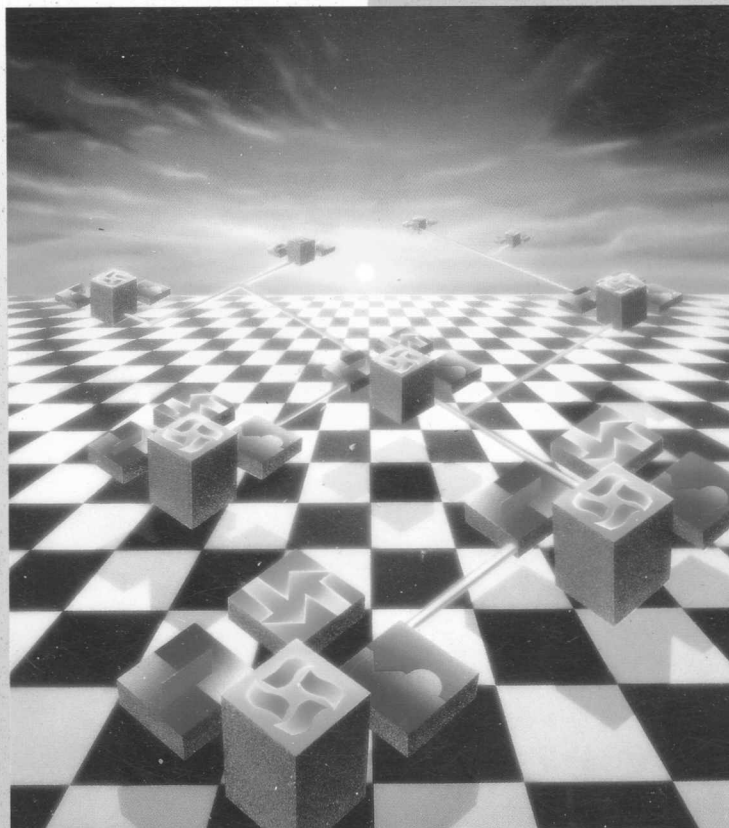
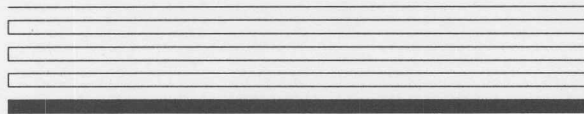


AK

LONBUILDER™ USER'S GUIDE



 ECHELON

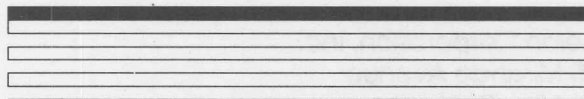


LONBUILDER™

User's Guide

Revision 2

ECHELON®
Corporation, Inc.



No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Echelon Corporation.

Echelon, LON, and NEURON are registered trademarks of Echelon Corporation. LONBUILDER, LONMANAGER, LONTALK, LONWORKS, 3120, and 3150 are trademarks of Echelon Corporation.

Document No. 29200

Printed in the United States of America.
Copyright ©1990, 1991, 1992 by Echelon Corporation

Echelon Corporation, Inc.
4015 Miranda Avenue
Palo Alto, California
94304

Preface

The LONBUILDER™ Developer's Workbench is an integrated hardware and software development environment running on an IBM PC-compatible host computer. The workbench includes all of the components required for the rapid development of LONWORKS™ applications.

Audience

The LONBUILDER User's Guide is for hardware and software developers, who are using the LONBUILDER Developer's Workbench to implement LONWORKS applications. Readers of this guide should have some programming knowledge or knowledge of basic digital hardware.

Purpose

The *LONBUILDER User's Guide* teaches developers how to use the LONBUILDER Developer's Workbench to develop and test LONWORKS applications.

Content

The *LONBUILDER User's Guide* has thirteen chapters and four appendices as follows:

- Chapter 1, *LONBUILDER Component Overview*, presents a comprehensive overview of the hardware and software components of the LONBUILDER Developer's Workbench.
- Chapter 2, *LONWORKS Application Development*, introduces the LONWORKS application development cycle.
- Chapter 3, *Introduction to the LONBUILDER User Interface*, explains how to use the LONBUILDER menus and windows.
- Chapter 4, *LONBUILDER System and Project Configuration*, explains how to create home and project directories and set system and project parameters.
- Chapter 5, *Editing Files*, describes the features of the LONBUILDER Editor.

- Chapter 6, *Defining Application Nodes*, describes how to define the application parameters for LONBUILDER NEURON Emulator, Single Board Computer (SBC), and custom nodes.
- Chapter 7, *Building Application Nodes*, describes how to use the LONBUILDER tools to compile, link, and load LONWORKS application programs. The chapter also describes how to build a custom node and export and import nodes and applications.
- Chapter 8, *Debugging Nodes*, describes how to use the NEURON C Debugger to debug from one to 24 nodes.
- Chapter 9, *Defining and Installing Management and Router Nodes*, describes how to define the hardware parameters of network management nodes and routers on a development network.
- Chapter 10, *Defining Development Networks*, describes how to define the network parameters of application nodes and routers on a development network.
- Chapter 11, *Creating Development Network Connections*, describes how to connect network variables and message tags and build a development network.
- Chapter 12, *Monitoring Development Networks*, describes how to use the LONBUILDER Protocol Analyzer to monitor and analyze a development network.
- Chapter 13, *Testing Development Networks*, describes how to use the LONBUILDER Network Manager to test the nodes on a development network.
- Appendix A, *Main Menu Bar*, defines the commands in the pull-down menus on the LONBUILDER main menu bar.
- Appendix B, *Editor Keyboard Summary*, summarizes the keystrokes that invoke LONBUILDER commands.
- Appendix C, *Sample Memory Map*, annotates a portion of a sample detailed memory map.
- Appendix D, *Utilities*, describes the various utility programs included with the LONBUILDER software.

Notices

The LONBUILDER Developer's Workbench uses the TesSeRact™ Ram-Resident Library and supports the TesSeRact Standard for Ram-Resident Program Communication. For information about TesSeRact, contact the TesSeRact Development Team at:

TesSeRact Development Team
1657 The Fairways
Suite 101
Jenkintown, PA 19046
1-215-884-3373

Compuserve: 70731,20
MCIMAIL: 315-5415

This MCIMAIL Account has been provided to the TesSeRact Development Team by Borland International, Inc. The TesSeRact Development Team is in no way associated with Borland International, Inc. TesSeRact is a trademark of the TesSeRact Development Team.

Contents

Preface	i
Audience.....	ii
Purpose.....	ii
Content	ii
Notices	iv

Part 1 LONBUILDER Overview and User Interface

1 LONBUILDER Component Overview.....	1-1
LONBUILDER Overview.....	1-2
LONBUILDER Hardware.....	1-3
LONBUILDER Development Station	1-3
LONBUILDER Processor Boards	1-5
LONBUILDER Neuron Emulator.....	1-6
LONBUILDER Single Board Computer (SBC).....	1-6
LONBUILDER Router	1-7
LONBUILDER Transceivers.....	1-7
LONBUILDER Twisted Pair Transceiver.....	1-8
LONBUILDER TP-RS485 Transceiver	1-9
LONBUILDER RF Transceiver.....	1-9
LONBUILDER I/O Expansion Boards.....	1-10

LONBUILDER Software.....	1-11
Application Programming Tools.....	1-11
Integrated Development Environment	1-12
NEURON C Developer's Kit.....	1-13
NEURON C Compiler.....	1-13
NEURON C Debugger	1-13
LONBUILDER Microprocessor Interface Developer's Kit.....	1-14
Network Management Tools.....	1-14
Network Manager	1-14
Protocol Analyzer.....	1-15
 2 LONWORKS Application Development.....	2-1
LONWORKS Application Development Cycle	2-2
Step 1: Define the Problem	2-5
Step 2: Identify Nodes and Assign Their Functions.....	2-5
Step 3: Define the External Interface for Each Node	2-5
Step 4: Write the Application Program for Each Node.....	2-5
Step 5: Debug and Test Individual Nodes.....	2-6
Step 6: Install Nodes in a Development Network and Test	2-8
 3 Introduction to the LONBUILDER User Interface	3-1
LONBUILDER Object Database	3-2
LONBUILDER Startup Screen	3-2
LONBUILDER Main Menu	3-3
Opening a Pull-Down Menu	3-4
Closing a Pull-Down Menu	3-5
Selecting a Command from a Pull-Down Menu.....	3-5
Submenus.....	3-6
Dialog Boxes	3-6
LONBUILDER Application Windows.....	3-8
Using Windows and Dialog Boxes	3-9
Object Lists	3-12
Buttons.....	3-13
Fields	3-13
Scroll Bars.....	3-15
Using the Navigator.....	3-17
Getting Help	3-19

4	LONBUILDER System and Project Configuration.....	4-1
	Setting Up Directories	4-2
	Home Directory	4-3
	Application Directories	4-4
	Project Directory	4-4
	Setting the LONPROJ Environment Variables	4-5
	Starting a LONBUILDER Session	4-5
	Configuring LONBUILDER System Parameters	4-7
	Configuring LONBUILDER Project Parameters	4-11
	Using a DOS Shell	4-15
	Ending a LONBUILDER Session	4-15
5	Editing Files.....	5-1
	Program Files	5-2
	Creating a Program File	5-2
	Modifying Program Files	5-3
	Deleting Program Files	5-4
	LONBUILDER Editor Window	5-5
	Basic Cursor Movement Commands	5-9
	Basic Editing Commands	5-14
	Prompt Editor	5-21
	Directory Display	5-22
	Using the Build Log with the Editor	5-23

Part 2 Node Implementation, Debug, and Test

6	Defining and Installing Application Nodes.....	6-1
	Application Node Definition	6-2
	Defining Properties	6-6
	Defining Memory Properties	6-10
	Defining Target Hardware	6-16
	Defining Application Images	6-20
	Defining Node Specifications	6-23
	Installing Application Node Target Hardware	6-26
	Modifying Application Node Objects	6-30
	Deleting Application Parameters	6-31

Standard Network Variable Types (SNVTs).....	6-32
The Node Query Window.....	6-32
Querying Network Variable Information.....	6-33
Editing Network Variable Names	6-35
Describing Network Variable Parameters.....	6-36
Importing SNV Information from the Node and	6-39
Preparing to Build	

7 Building Application Nodes	7-1
Selecting a Project Build Command.....	7-2
Compiling NEURON C Programs.....	7-6
Loading Built Nodes and Routers.....	7-8
Loading Application Nodes and Routers Automatically	7-9
Loading Application Nodes Manually.....	7-9
Loading Routers Manually	7-10
Loading Nodes Through Routers	7-10
Building Custom Nodes.....	7-9
Exporting and Importing	7-12
The NEURON ROM Image (.NRI) File	7-13
The NEURON EEPROM Image (.NEI) File	7-13
The Downloadable Image (.NXE) File.....	7-14
The External Interface (.XIF) File	7-14
The NEURON Object and Connection Information Files	7-15
The Components of an Exported Node.....	7-15
Exporting Node Files.....	7-17
Importing External Interface Files	7-21
Exporting Application Image Files.....	7-22
Importing Application Image Files	7-23
Building Custom Nodes.....	7-26
Target Network Types and Firmware State Selection	7-29
Programming Custom Nodes.....	7-31
Programming NEURON 3150 CHIP Memory.....	7-31
Programming NEURON 3120 CHIP Memory.....	7-32
Installing Custom Nodes.....	7-36
Loading Custom Nodes	7-37
Using Multiple Firmware Versions.....	7-38
Modifying Firmware Settings	7-38

8	Debugging Nodes	8-1
	NEURON C Debugger	8-2
	Debugger Window	8-2
	Read and Write Protect	8-5
	Debugger Commands	8-6
	Debugger File Markers	8-7
	Running and Debugging Programs	8-7
	Running and Debugging Code	8-8
	Halting Running Nodes	8-9
	Setting Breakpoints	8-9
	Single Stepping Over Functions	8-11
	Trace Stepping Into Functions	8-12
	Displaying the Current Call Sequence	8-12
	Changing the Current Context	8-14
	Evaluating and Modifying Variables	8-16
	Evaluating a Variable	8-16
	Modifying a Variable	8-20
	Evaluating and Modifying Raw Memory	8-23
	Network Variables	8-25
	Timers	8-26
	Pointer Variables	8-26
	Addresses of Static Variables	8-28
	Using the Program Error Log	8-29

Part 3 LONBUILDER Network Installation

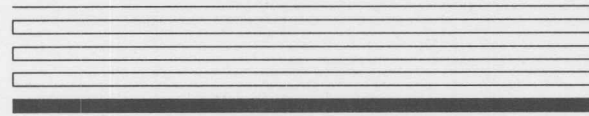
9	Defining and Installing Network Management and Router Nodes	9-1
	Development Network Integration	9-2
	Network Management Node Target Hardware	9-5
	Installing Network Management Node Target Hardware	9-6
	Defining Multi-Channel Topologies	9-8
	Topology Errors	9-9
	Defining Router Target Hardware	9-12
	Installing Router Target Hardware	9-18
	Defining Router Node Specifications	9-22

10 Defining Development Networks.....	10-1
Loading Nodes in a Development Network.....	10-2
Defining Simple Development Networks.....	10-3
Defining Node Specifications.....	10-4
Defining Large Development Networks.....	10-8
Defining Domains.....	10-12
Defining Subnets.....	10-14
Defining Channels.....	10-16
Specifying Custom Transceiver Types.....	10-21
Copying Network Objects.....	10-29
Modifying Network Objects.....	10-30
Deleting Network Objects.....	10-31
 11 Creating Development Network Connections	11-1
Connections	11-2
Creating Connections.....	11-5
Modifying Connections.....	11-7
Deleting Connections.....	11-9
Setting Parameters for Connections	11-10
Building the Network Image.....	11-15
Node Configuration Reports.....	11-16
Creating a Node Configuration Report.....	11-20
Viewing a Node Configuration Report.....	11-21
 12 Monitoring Development Networks	12-1
LONBUILDER Protocol Analysis Tools.....	12-2
Viewing Network Traffic Statistics.....	12-2
Collecting and Viewing Message Packets.....	12-6
Creating Packet Logs	12-6
Turning on a Packet Log.....	12-7
Defining the Packet Log Filter.....	12-8
Viewing a Packet Log.....	12-9
Network Management Message Interpretation.....	12-17

13 Testing Development Networks.....	13-1
LONBUILDER Network Manager Test Commands.....	13-2
Testing Application Nodes	13-6
Testing Routers	13-9
Taking Application Nodes On and Off Line	13-10
Taking Routers On and Off Line.....	13-11
Resetting Nodes.....	13-11
Resetting Routers	13-12
Winking Nodes.....	13-12
Browsing Node Memory	13-13
Multi-Channel Network Management Commands.....	13-15
The Network Variable Browser.....	13-16
Using the Network Variable (NV) Browser.....	13-16
Viewing Data in a Browser List	13-18
Modifying Network Variables in a Browser List.....	13-20
Adding and Deleting Variables in a Browser List.....	13-22
Updating Data in a Browser List.....	13-23

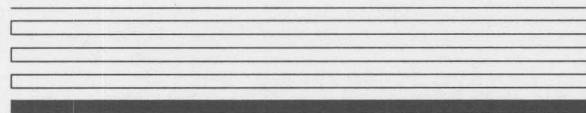
Part 4 Appendices

A Main Menu Bar	A-1
Menu Commands.....	A-2
System Menu	A-5
File Menu.....	A-7
Edit Menu.....	A-10
Search Menu.....	A-13
View Menu.....	A-17
Options Menu.....	A-20
Project Menu	A-25
Macro Menu.....	A-27
Window Menu	A-30
Help Menu.....	A-33
 B Keyboard Summary	 B-1
 C Sample Detailed Memory Map	 C-1
 D Utilities.....	 D-1



Part 1

LONBUILDER Overview and User Interface



1

LONBUILDER Component Overview

This chapter presents a comprehensive overview of the hardware and software components of the LONBUILDER Developer's Workbench.

LONBUILDER Overview

The LONBUILDER Developer's Workbench is a development environment providing the tools needed to create LONWORKS applications. A LONWORKS application consists of intelligent devices, or nodes, that are connected by one or more communications media and that communicate with one another using the LONTALK™ protocol. The LONBUILDER system is used to program nodes to send messages to one another in response to changes in various conditions, and to take action in response to messages they receive. Each node contains:

- A NEURON™ CHIP - a programmable device that implements the LONTALK protocol and performs the node's intended application function.
- A transceiver to provide the mechanical and electrical interface between the node's NEURON CHIP and the communications media.
- Circuitry to connect the NEURON CHIP to I/O devices such as actuators, display devices, etc.

The LONBUILDER system facilitates the independent development of individual nodes whose function may be simple or complex, and integration of these nodes into LONWORKS applications which might perform complex and sophisticated tasks.

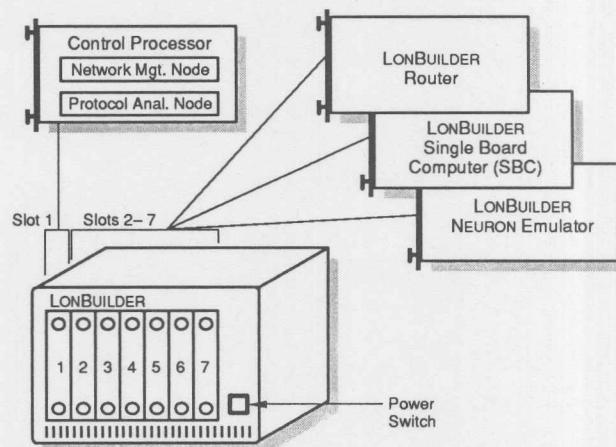
LONBUILDER Hardware

The LONBUILDER Developer's Workbench consists of integrated hardware and software that operates with an IBM PC/AT-compatible PC. During development, a LONWORKS application typically grows from a network of two communicating nodes to a network of many nodes. The LONBUILDER Developer's Workbench accommodates expandability with the LONBUILDER development station, LONBUILDER processor boards, and the LONBUILDER expansion boards. On a single PC, the LONBUILDER system provides a development environment that can grow from a pair of emulated application nodes, residing in the development station, to a distributed system of up to 24 emulated nodes and 256 remote nodes.

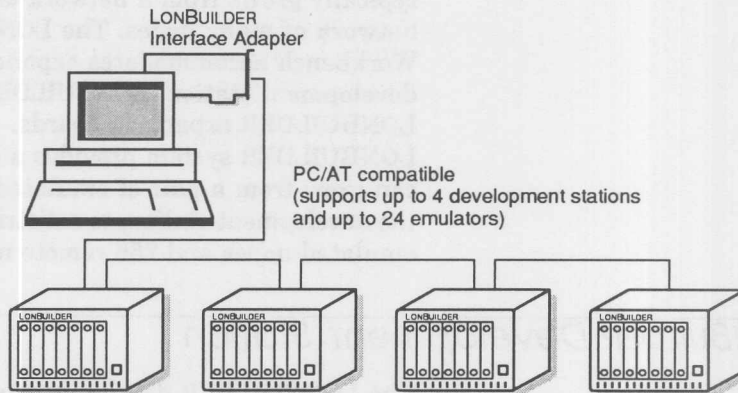
LONBUILDER Development Station

The LONBUILDER development station includes:

- An enclosure with power supply and backplane hardware that supports up to six optional processor boards.
- A control processor to support communication between the development station and the processor boards. The control processor communicates with the PC through the LONBUILDER Interface Adapter, and includes a network management node and a protocol analysis node.



A LONBUILDER Interface Adaptor installed in the host PC provides high speed communications between the PC software and the processor boards in the development station. Up to four development stations can be connected to a single PC, supporting a total of up to 24 processor boards.



LONBUILDER Processor Boards

There are three types of LONBUILDER processor boards: LONBUILDER NEURON Emulators, LONBUILDER Single Board Computers (SBC), and LONBUILDER Routers.

Processor boards may accept two types of optional expansion boards: transceiver and I/O. The optional expansion boards can be LONBUILDER transceivers, LONBUILDER I/O boards, custom transceivers, or custom I/O interfaces.

Table 1-1 lists the components of the LONBUILDER processor boards.

Table 1-1. LONBUILDER Processor Board Components

Processor Board	Processors	Amount of Memory	Accepts Optional Transceiver Expansion Boards	Accepts Optional I/O Expansion Boards	Supports the NEURON C Debugger
Emulator	one NEURON 3150 CHIP	64 KB application 64 KB control	yes one board	yes one board	yes
SBC	one NEURON 3150 CHIP	64 KB application	yes one board	yes one board	no
Router	two NEURON 3150 CHIPS	32 KB system per NEURON 3150 CHIP	yes two boards	no	no

LONBUILDER NEURON Emulator

Nodes are initially developed on the LONBUILDER NEURON Emulator, a processor board that allows source-level software debugging using the NEURON C Debugger. The emulator incorporates a NEURON 3150™ CHIP with 64 KB of RAM. It also includes an additional 64K of memory for use by the debugger in controlling the application program. The emulator accepts up to two optional expansion boards for testing with prototype transceiver and I/O hardware.

The emulator always resides in the development station, where it provides hardware support for application loading, source-level breakpoints, single-stepping, reset/start/stop, and memory read/write protection. The emulator also provides a software controlled clock rate which may be set to 10 MHz, 5 MHz, 2.5 MHz, 1.25 MHz, and 625 KHz. The LONBUILDER software can emulate the NEURON 3120™ CHIP memory map on the emulator's NEURON 3150 CHIP.

LONBUILDER Single Board Computer (SBC)

The LONBUILDER SBC is a single-board computer incorporating a NEURON 3150 CHIP with 64 KB of nonvolatile RAM. After debugging application software on the emulator, it can be moved to an SBC. The SBC can be installed in the development station or used remotely with an external power supply. The SBC accepts the same expansion boards as the emulator, allowing you to test prototype transceiver and I/O hardware in a remote node.

LONBUILDER Router

The LONBUILDER Router is a processor board incorporating two NEURON 3150 CHIPS. Each router supports two optional transceiver expansion boards to provide routing between two communications media or channels. Routers may be installed as follows:

- Learning router, which monitors network traffic to learn the network topology. The router uses the extracted network topology information to selectively route packets between channels.
- Configured router, which uses routing tables based on the network configuration of the nodes to route packets between channels. The LONBUILDER Network Manager automatically configures the routing tables.
- Bridge, which forwards all packets between the two connected channels on the domains in which the bridge is installed.
- Repeater, which forwards all packets between the two connected channels, regardless of the domain in which the repeater is installed.

A LONBUILDER Router may be installed in the development station or used remotely with an external power supply.

LONBUILDER Transceivers

Processor boards require a transceiver expansion board to provide the physical interface to a communications channel. LONBUILDER transceivers are available for twisted pair and radio frequency networks. Custom transceivers may also be developed.

All processor boards include a backplane transceiver that can be used to install the board on the backplane network within a development station. Configurable bit rates are 4.8 Kbps, 9.8 Kbps, 19.5 Kbps, 39.1 Kbps, 78.1 Kbps, 156.3 Kbps, 312.5 Kbps, 625 Kbps, and 1.25 Mbps. Collision detection requires a LONBUILDER Twisted Pair Transceiver and is not provided on the backplane network.

The backplane network is limited to the boards contained within a given development station: a maximum of 6 processor boards. When you exceed the capacity of the backplane network or when you're ready to test your nodes on target media, you can add or change transceiver expansion boards to change the network media, usually without affecting the application source code.

LONBUILDER Twisted Pair Transceiver

The LONBUILDER Twisted Pair Transceiver provides the physical interface to a LONWORKS TP/XF-78 or TP/XF-1250 twisted pair network. This transceiver includes circuitry supporting interfaces for two bit rates: 78 Kbps and 1.25 Mbps. You can use this transceiver to extend development networks on the twisted pair medium to the number of nodes and distances shown in table 1-2.

Table 1-2. LonBuilder Twisted Pair Transceiver Performance

<i>Data Rates</i>	<i>Max Number of Nodes Per Channel</i>	<i>Stub Length</i>	<i>Bus Length</i>
78 Kbps	64 Node Equivalents ¹	3 meters (10 feet)	2000 meters (6560 feet)
1.25 Mbps	64 Node Equivalents ¹	0.3 meters (1 foot)	500 meters (1640 feet)

Each node that uses a LONBUILDER Twisted Pair Transceiver counts as two nodes. If all LONBUILDER Twisted Pair Transceivers are used, the maximum number of nodes per twisted pair channel is 32.

Note that the number of nodes supported in the LONBUILDER development environment with the twisted pair transceiver is somewhat less than the full capability of the LONWORKS twisted pair interface specifications.

To ensure the flexibility of snap-on transceivers in the development environment, each LONBUILDER twisted pair transceiver places two loads on the network. Consequently, when all LONBUILDER Twisted Pair Transceivers are used, the maximum number of nodes is 32. The LONBUILDER Twisted Pair Transceiver is explained more fully in the *LONBUILDER Startup and Hardware Guide* and the *Implementing Twisted Pair Transceivers with NEURON CHIPS* engineering bulletin.

LONBUILDER TP-RS485 Transceiver

The LONBUILDER TP-RS485 Transceiver provides the physical interface to a LONWORKS TP-RS485 twisted pair network based on the EIA RS-485 standard. The TP-RS485 transceiver supports standard LONWORKS bit rates: 4.8 Kbps, 9.8 Kbps, 19.5 Kbps, 39.1 Kbps, 78.1 Kbps, 156.3 Kbps, 312.5 Kbps, 625 Kbps, and 1.25 Mbps. The recommended bit rate is 39.1 Mbps. See the *LONBUILDER Startup and Hardware Guide* and the *Implementing RS-485 Transceivers* engineering bulletin for more information. The bus length supported by the TP-RS485 transceiver is as specified in the EIA RS-485 standard. At 39.1 Kbps, the bus length is 1200m (3935 feet).

LONBUILDER RF Transceiver

The LONBUILDER RF Transceiver provides the physical interface to a radio frequency network. This transceiver provides a 4822 bps bit rate using a 49 MHz carrier. The RF transceiver is licensed under FCC part 15 for use in North America. Certain restrictions apply. See the *LONBUILDER Startup and Hardware Guide* for details.

LONBUILDER I/O Expansion Boards

The LONBUILDER processor boards accept a second expansion board in addition to a transceiver expansion board. The second board is used for prototyping I/O interfaces. There are several options for developing I/O interface prototypes:

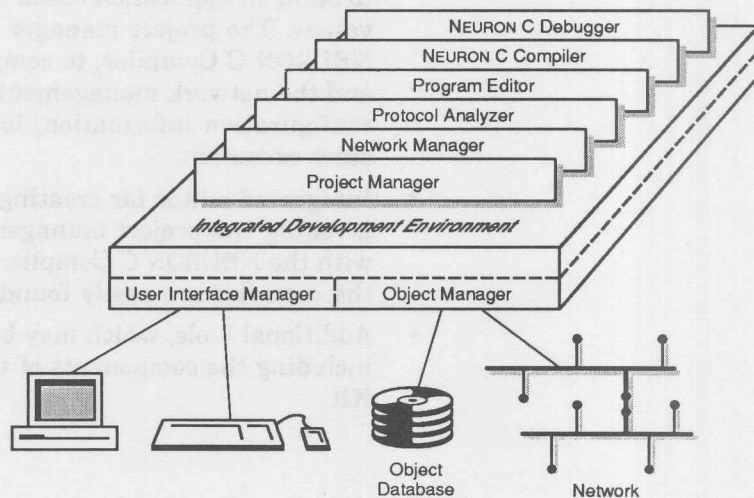
- Build a custom I/O expansion board as described in the *LONBUILDER Startup and Hardware Guide*.
- Use the LONBUILDER I/O Evaluation Board to develop prototype I/O interfaces. The I/O evaluation board includes buffers for the NEURON CHIP I/O signals and a large prototyping area for developing prototype hardware. The LONBUILDER Extender Card simplifies access to hardware implemented on an I/O evaluation board.
- Use the LONBUILDER Application Interface Board to interface with I/O devices and custom node hardware external to the development station. The LONBUILDER Multi-Function I/O Kit includes an application interface board and a Gizmo 2 multi-function I/O module providing a collection of I/O devices useful for prototyping LONWORKS applications. You can use the Gizmo 2 for initial software development, then replace it with your prototype hardware when the hardware becomes available.

LONBUILDER Software

LONBUILDER software tools provide application programming tools familiar to developers of microcontroller-based applications, as well as network management tools that meet the unique needs of LONWORKS application development.

Application Programming Tools

The LONBUILDER application programming tools include the project manager, editor, compiler, and source-level debugger. All of these tools are linked through an Integrated Development Environment (IDE), as shown in the following diagram.



Integrated Development Environment

The LONBUILDER Integrated Development Environment (IDE) supports application programming by automating the development cycle and providing a common framework for the software tools. The basic IDE includes the following:

- Object database, which stores the values of the user-definable parameters of application, router, and network management nodes. These values are shared by all the LONBUILDER software tools.
- Navigator, which simplifies viewing and changing the object database. A large project may consist of hundreds of objects. The definitions of these objects are easily managed using the navigator.
- Project manager, which manages the configuration of nodes defined within the object database and enables you to build an application based on the object database values. The project manager invokes the optional NEURON C Compiler, to compile application programs, and the network management tools to build the network configuration information, load the application, and start execution.
- Integrated editor for creating application programs and invoking the project manager. The editor is integrated with the NEURON C Compiler, so that errors identified by the compiler are easily found and corrected.
- Additional tools, which may be added to the basic IDE, including the components of the NEURON C Developer's Kit.

NEURON C Developer's Kit

→ now part
of LONBUILDER
V2.2.

The optional NEURON C Developer's Kit includes the NEURON C Compiler and NEURON C Debugger.

NEURON C Compiler

The NEURON C Compiler is a cross compiler, taking as input NEURON C source code stored on the PC, and generating code that is stored in the object database for subsequent loading to one or more target nodes. The NEURON C Compiler generates code for both the NEURON 3120 CHIP and the NEURON 3150 CHIP.

NEURON C Debugger

The NEURON C Debugger is a cross debugger running on a PC host while debugging NEURON C applications running on one to 24 NEURON emulators. The NEURON C Debugger provides a full-screen source-level view of application programs executing on NEURON emulators. You use the source-level views to set breakpoints, start and stop program execution, and single-step through the programs. You can also evaluate and modify program variables using NEURON C symbolic names. The NEURON C Debugger provides visibility into the program's call stack by providing commands to display the call stack and change the current context to any function within the stack.

LONBUILDER Microprocessor Interface Developer's Kit

The optional LONBUILDER Microprocessor Interface Developer's Kit enables you to turn the NEURON CHIP into a communications chip that can be used with any other processor. The kit includes the Microprocessor Interface Program (MIP), a NEURON CHIP application that exports the top layers of the LONTALK protocol from the NEURON CHIP to the external host processor. For faster development, the Serial LONTALK Adapter provides a ready to use serial interface to a LONWORKS network.

Network Management Tools

The LONBUILDER network management tools include the LONBUILDER Network Manager and the LONBUILDER Protocol Analyzer. These tools are used to define, configure, load, monitor, and control multiple nodes in a development network.

Network Manager

The LONBUILDER Network Manager provides the tools to define, configure, load, and control LONWORKS nodes and networks. The network manager uses the network interface included in the development station control processor to send network management commands to other nodes within the network.

The network manager is used to accomplish the following:

- Define the physical configuration of nodes by associating target hardware with an application source program. This defines the *application image*.
- Define network connections, generate node addressing information (called the *network image*), and load application and network images on target hardware.
- Put nodes on-line, take nodes off-line, reset nodes, read node memory, write node memory, wink, and test nodes.
- Monitor the status of all nodes on the network.
- Examine and change network variables on any node in the network.
- Retrieve self-identification and self-documentation data describing the network variables in any node.

Protocol Analyzer

The LONBUILDER Protocol Analyzer selectively monitors, collects, and displays network traffic and network performance statistics. The protocol analyzer uses the packet monitor node included in the control processor to monitor network traffic.

The protocol analyzer is used to accomplish the following:

- View network traffic.
- Save all network traffic to a log file for later display.
- Save specific network traffic to a log file by specifying packet filters based on source node, destination node, packet type, network variable, and message tag.
- Summarize network performance, packet counts by packet type, and error counts.

2

LONWORKS Application Development

This chapter presents an overview of the LONWORKS application development cycle.

LONWORKS Application Development Cycle

Your LONWORKS application might consist of a single node, incorporate several nodes embedded within it, or be a collection of nodes installed in the field in a custom LONWORKS network. Regardless of the overall complexity, the development cycle of a LONWORKS application typically includes these steps:

- Step 1: Define the problem
- Step 2: Identify nodes and assign their functions
- Step 3: Define the external interface for each node
- Step 4: Write the application program for each node
- Step 5: Debug and test individual nodes
- Step 6: Install nodes in a development network and test

LONWORKS applications are initially developed using prototype nodes installed on a development network. The development network serves as a prototype of the networks that will ultimately be created with the new application. The LONBUILDER Developer's Workbench provides a number of special tools that simplify the definition and management of the development network. Although different tools may be used to install LONWORKS application nodes in the field, the installation steps are generally the same as the steps for installing application nodes on a development network.

A node's memory consists of three images: (1) system image, (2) application image, and (3) network image. A node's system image includes its NEURON CHIP firmware. The system image is fixed for all NEURON CHIPS of a given type.

The application image and network image are the user-definable portions of the node's memory. A node's application image includes its application program and its hardware, I/O, and transceiver configuration information. Many nodes in a LONWORKS network might have the same application image. For example, in a factory automation system, all conveyor belt motor nodes could contain the same application image: they could have the same application program and the same hardware, I/O, and transceiver configuration information.

A node's network image defines its relationship to other nodes in the network. The network image, stored in the EEPROM of the NEURON CHIP, gives a node its unique behavior in the network. The network image allows one conveyor belt motor node to behave differently from the next when installed in a network.

During development, you use the LONBUILDER Developer's Workbench to create a node's application image in steps 2 through 5 of the development cycle and to create its network image in step 6 of the development cycle. During production, a node is typically manufactured with its system and application image. The network image may also be installed during production, or may be installed in the field. When installed in the field, a network management tool may be used to create and load a network image on the node. In addition, an application image can be loaded when the node is installed in the field. See the *Installing LONWORKS Installation Overview* engineering bulletin and the *NEURON CHIP-based Installation of LONWORKS* engineering bulletin for more information on LONWORKS installation.

Figure 2-1 charts the LONWORKS application development cycle. The remainder of this chapter describes the tasks for each step in the cycle.

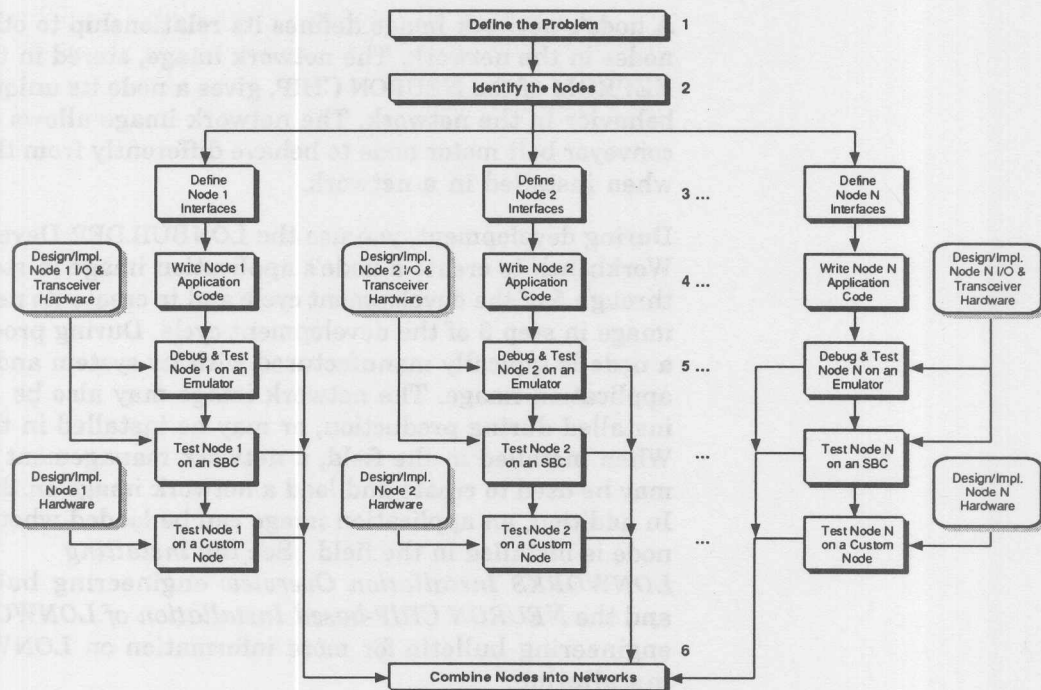


Figure 2-1. LONWORKS Application Development Cycle

Step 1: Define the Problem

During the problem definition step of the development cycle, you define the desired functionality of the product. This step is described in the *NEURON C Programmer's Guide*.

Step 2: Identify Nodes and Assign Their Functions

During the node identification and function assignment step of the development cycle, you partition the application into one or more nodes. Each node is an independent object that consists of a NEURON CHIP, one or more I/O devices, and a communications transceiver. This step is described in the *NEURON C Programmer's Guide*.

Step 3: Define the External Interface for Each Node

During the node interface definition step of the development cycle, you define the interfaces of the node that will be visible to other nodes. These interfaces are defined using application-level network variables and explicit messages. The network variables and explicit messages of a node are the only part of the node that is *visible* to other nodes. Defining nodes in terms of their external interfaces allows them to be developed independently, and minimizes the impact of network or application changes. This step is described in the *NEURON C Programmer's Guide*.

Step 4: Write the Application Program for Each Node

During the programming step of the development cycle, you write application programs to implement the desired functionality for each node.

Application programs may be written in NEURON C, using the NEURON CHIP as both the applications processor and the communications processor.

Application programs may also be created for other host processors using the LONBUILDER Microprocessor Interface Program (MIP). The MIP moves the applications processing from the NEURON CHIP to an external host processor.

For NEURON C applications, this step is described in the *NEURON C Programmer's Guide*. For the development of host-based applications, see the *LONWORKS Host Application Programmer's Guide*.

Step 5: Debug and Test Individual Nodes

During the debugging step of the development cycle, you use the LONBUILDER Developer's Workbench to perform these tasks for each application node:

- 1 *Install and configure* a LONBUILDER NEURON Emulator in a development station.
- 2 *Compile* and link application code for the node, and load the application program onto the installed and configured emulator.
- 3 *Debug* the node's application program running on the emulator using the NEURON C Debugger.

Repeat these tasks for each of the application's nodes. At your option, you can choose to:

- Use the same emulator to debug the application code for each node.
- Attach a LONBUILDER transceiver or custom transceiver to the emulator to test the node on target media.
- Add a LONBUILDER I/O board or custom I/O interface to the emulator to test prototype I/O devices.

Optionally, following debugging on an emulator, you may test application code on an SBC. The tasks let you:

- 4 *Install and configure* a remote LONBUILDER SBC on the target media using a LONBUILDER or custom transceiver. (SBCs must be initially installed within the development station prior to remote use.)
- 5 *Load* the node's application program onto the installed and configured SBC.
- 6 *Test* the node's application program running on the SBC. The LONBUILDER Network Manager is used to install, load and control remote SBCs. The NEURON C Debugger cannot be used with SBCs.

These tasks may be repeated for each of the application's nodes.

Also at your option, following debugging on an emulator and testing on an SBC, you can test the node's application program on your own custom hardware. The tasks let you:

- 7 *Install and configure* your custom node on the target medium. Custom nodes may be based on either LONWORKS Control Modules, or on custom designs.
- 8 *Load* the node's application program onto the installed and configured custom node.
- 9 *Test* the node's application program running on the custom node. The LONBUILDER Network Manager is used to load the writable portion of the application image and test custom nodes. The NEURON C Debugger cannot be used with custom nodes.

These tasks may be repeated for each of the application's nodes.

The tasks to accomplish this step are described in Part 2 of this guide.

Step 6: Install Nodes in a Development Network and Test

During the network installation step of the development cycle, use the LONBUILDER Network Manager to install two or more emulator, SBC, or custom nodes in a development network. The installation tasks include:

- 1 *Application definition:* to define the functionality of the nodes in the development network. The functionality can change as you add more nodes to the network.
- 2 *Development network design:* to determine the number and type of nodes, how the nodes will be logically connected to each other, and where the nodes will be physically installed.
- 3 *Physical placement and attachment:* to locate nodes in their proper places and to make any necessary attachment to application hardware and to network media.
- 4 *Node customization:* to load nodes with information that establishes the desired logical connections to other nodes.
- 5 *Network test:* to monitor and test communication among the nodes on the development network.

The development network integration step is cyclic. Repeat the tasks as you add nodes to the network.

The tasks to accomplish this step are described in Part 3 of this guide.

3

Introduction to the LONBUILDER User Interface

This chapter introduces the LONBUILDER user interface. It describes how to:

- Open menus and windows.
- Use a mouse, arrow keys, and keyboard to select from menus and windows.
- Move from window to window.
- Use the LONBUILDER Navigator.
- Get help.

LONBUILDER Object Database

As you use the LONBUILDER system, you create objects that define your application. The objects include:

- Application programs
- Node definitions
- Development network configuration information

The node definitions and network configuration information are stored in an object database. This object database is shared by all the LONBUILDER development tools, ensuring each tool the same view of your application.

LONBUILDER Startup Screen

After issuing the command to start the LONBUILDER software, as described in Chapter 4, the main menu bar, the Navigator home window, and the status line are displayed. Figure 3-1 shows the startup screen.

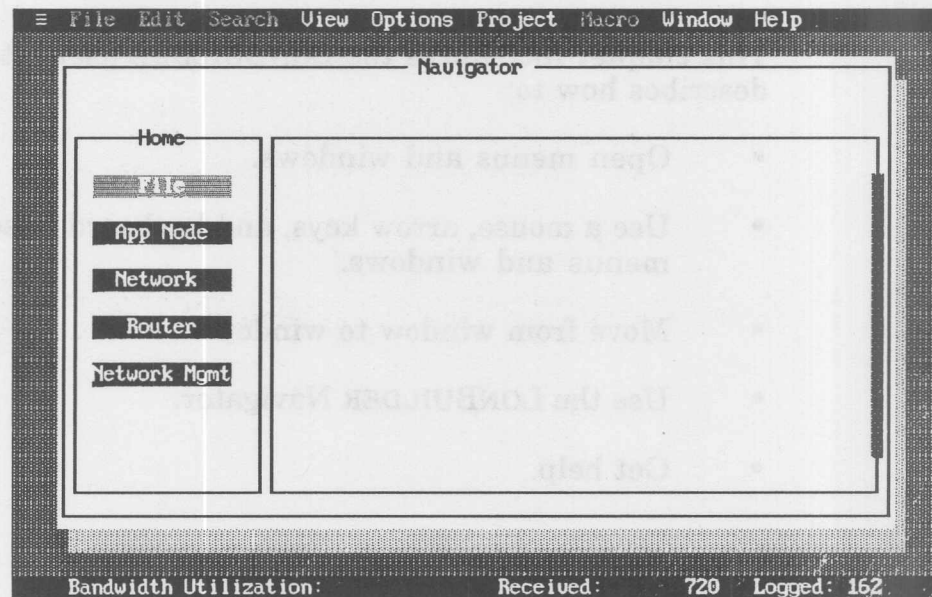


Figure 3-1. Navigator Home Window

The main menu bar provides pull-down menus from which you select a command that either opens a new window, opens a dialog box, or performs a system or editor operation. The main menu bar and how to use it are described under *LonBuilder Main Menu*, later, in this chapter.

The Navigator window is one of six LONBUILDER application windows, which provide the interface to the LONBUILDER development tools. You use the Navigator window to *navigate* through the object database creating and changing your application's objects. You can even open the other application windows from the Navigator window. The Navigator and how to use it are described under *Using the Navigator*, later, in this chapter.

The status line summarizes information from the protocol analyzer, including the bandwidth utilization of the communications medium, the number of packets received, and the number of packets logged. See Chapter 12, *Monitoring Development Networks*, for a description of the fields in the status line.

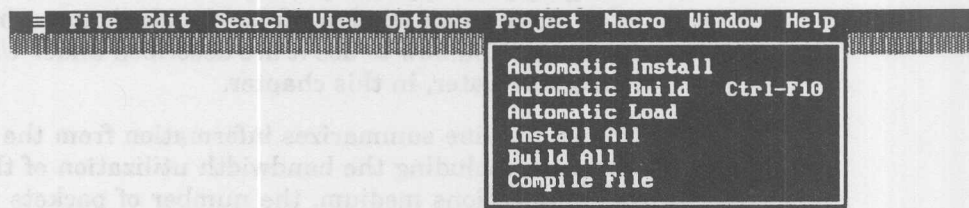
LONBUILDER Main Menu

The menu bar provides pull-down menus from which you select commands. Available pull-down menus are highlighted and change based on the open application window.



These commands are described in Appendix A, Main Menu Bar.

Opening a menu and selecting a command from it causes an action to occur: some commands open application windows; some open dialog boxes; some perform system or editor operations.



Opening a Pull-Down Menu

Accelerator keys are defined in Appendix A, Main Menu Bar, and summarized in Appendix B, Keyboard Summary. Many accelerator keys require you to hold down the CONTROL, ALT, or SHIFT key while you press another key. The notation for these accelerator keys is Ctrl-key, Alt-key, Shift-key.

This chapter uses the word *mouse* to mean any Microsoft mouse compatible pointing device.

To open a pull-down menu, either:

- 1 Type the accelerator key that selects the menu. (Accelerator keys are single-key commands that provide a short cut for entering commands. Accelerator keys for the pull-down menus are always ALT combined with the highlighted letter of the menu command; for example: *Alt-F* for File, *Alt-E* for Edit.)
- 2 If you are not in the editor window, press *Enter* (the ENTER key). The menu opens with the first command highlighted.

or

- 1 Use the mouse to move the pointer to the menu title.
- 2 Click the mouse button. The menu opens with the first command highlighted.

or

- 1 Press *F10* (the F10 function key) to highlight the most recently selected pull-down menu.
- 2 Use the left and right arrow keys to highlight the pull-down menu you want to open.
- 3 Press Enter. The menu opens with the first command highlighted.

or

- 1 Press *F10* (the F10 function key) to highlight the most recently selected pull-down menu.
- 2 Type the highlighted letter of the title of the pull-down menu you want to open. The menu opens with the first command highlighted.

Closing a Pull-Down Menu

After you've opened a pull-down menu, you can close it without selecting a command by pressing *Esc* (the ESCAPE key). Press *Esc* again if a pull-down menu is still highlighted. Or, click anywhere outside of the pulldown menu to close the currently opened pull-down menu.

Selecting a Command from a Pull-Down Menu

Accelerator keys for the commands on the pull-down menus are displayed to the right of the command.

The quickest way to invoke a command on a pull-down menu is to type its accelerator key. The pull-down menu does not need to be open for the accelerator key to work.

To select a command from an open pull-down menu, either:

- 1 Use the mouse to move the pointer to the command.
- 2 Click the mouse button. Clicking the button invokes the command.

or

- 1 Use the down arrow key or up arrow key to highlight the command.
- 2 Press *Enter*. Pressing the ENTER key invokes the command.

or

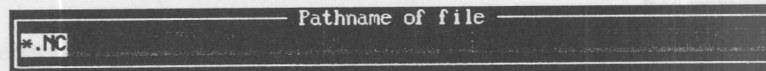
Type the highlighted letter of the command.

Submenus

On the pull-down menus, some of the commands that end in an ellipsis (...) open a submenu. Select from the submenu just as you would from a pull-down menu. To close an open submenu without selecting a command, press *Esc* or click anywhere outside the submenu.

Dialog Boxes

On the pull-down menus, some of the commands that end in an ellipsis (...) open a dialog box. A dialog box is a window that requests information or requires a response. Some dialog boxes, such as the one pictured below, are very simple. This dialog box prompts you for a filename. Press *Enter* after entering the filename for your response to take effect. Press *Esc* to close the dialog box without responding.



Some dialog boxes, such as the one in figure 3-2, are more extensive, with the same properties as LONBUILDER application windows. These dialog boxes usually have an OK or SAVE button to save the information you enter and a CANCEL button to close the box without saving. If there is no CANCEL button, pressing *Esc* closes the dialog box. (See *Using Windows and Dialog Boxes*, later.)

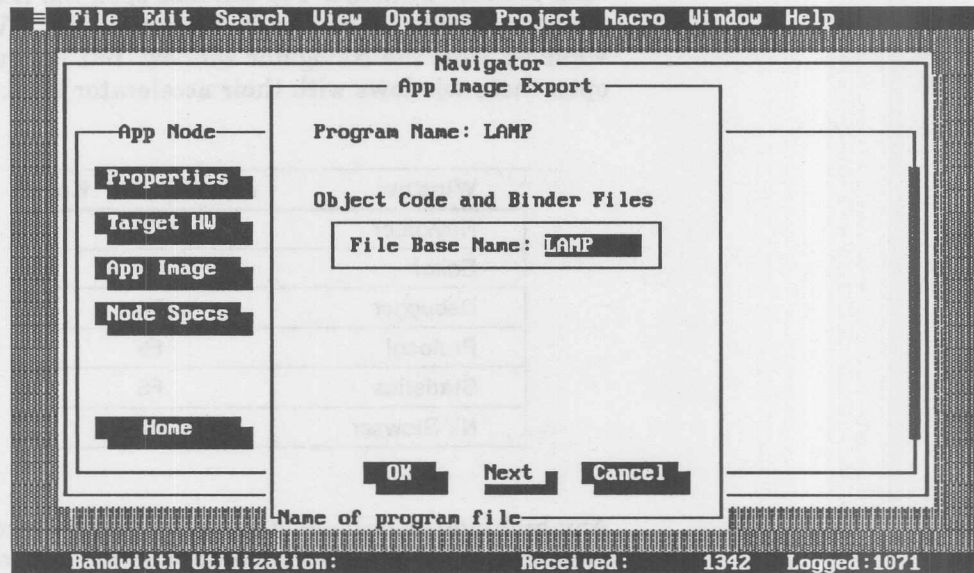


Figure 3-2. Sample LONBUILDER Dialog Box

When a dialog box opens, the main menu bar and any other open windows are unavailable until you close the dialog box.

LONBUILDER Application Windows

There are six LONBUILDER application windows: Navigator, Editor, Debugger, Protocol Analyzer, Statistics, and the Network Variable Browser (NV Browser). These windows are the interface to the LONBUILDER development tools. The Window pull-down menu has a command to open each of these windows. You can also open the Editor, Debugger, Protocol Analyzer, Statistics, and NV Browser windows from the Navigator window. But it's quickest to open these windows with their accelerator keys.

Window	Accelerator Key
Navigator	F2
Editor	F3
Debugger	F4
Protocol	F5
Statistics	F6
NV Browser	F7

Any highlighted pull-down menu on the main menu bar is available to use from an open application window. The available pull-down menus vary depending on which application window is open. However, from an open application window, you can always use an accelerator key or the window pull-down menu to open another application window. This action closes the current window, but saves its state should you want to return to it.

Using Windows and Dialog Boxes

LONBUILDER windows consist of fields, object lists, buttons, and scroll bars. The window in figure 3-3 has subwindows of buttons and a display list.

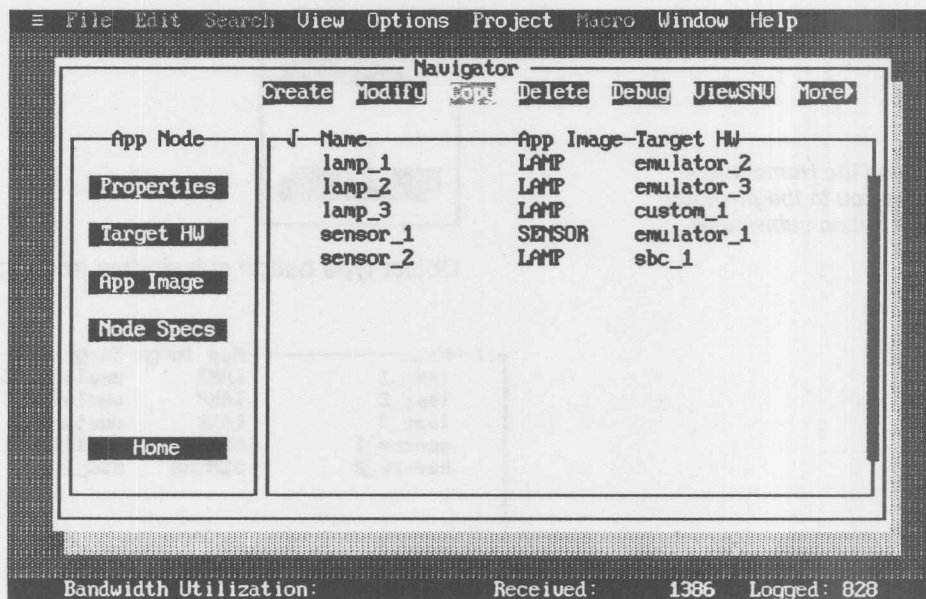


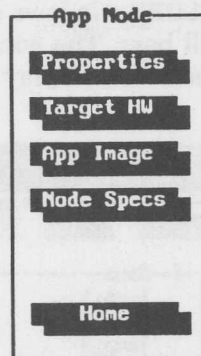
Figure 3-3. Sample LONBUILDER Window



Pressing the More button displays additional command buttons in the subwindow.

Command button subwindow from figure 3-3

Pressing the Home button returns you to the previous object button subwindow.



Object-type button subwindow from figure 3-3

J-Name	App Image	Target HW
lamp_1	LAMP	emulator_2
lamp_2	LAMP	emulator_3
lamp_3	LAMP	custom_1
sensor_1	SENSOR	emulator_1
sensor_2	SENSOR	SBC_1

Object list subwindow from figure 3-3

The window in figure 3-4 contains fields in which you enter values, fields from which you choose a list of possible values, fields that open a menu from which you select a value, and read-only fields.

≡ File Edit Search View Options Project Macro Window Help

Node Modify

Node Name: App Image Name: ↓

Topology

Subnet 1: Subnet 2: ↓

Auth Key 1: Auth Key 2:

Messaging	Hardware
Net Mgt Authenticate: <input type="text" value="No"/>	Target HW: <input type="text" value="emulator_2"/> ↓
Max Free Buffer Wait: <input type="text" value="10"/> s	Location: <input type="text" value=""/>

Node name must be unique

Bandwidth Utilization: Received: 1492 Logged: 934

Figure 3-4. Sample LONBUILDER Window

This section presents generic instruction on how to use the elements of LONBUILDER windows and dialog boxes. The individual LONBUILDER windows and dialog boxes and how to use them to accomplish development tasks are described in Chapters 5 through 13 of this guide.

Object Lists

To select an item from an object list, either:

- 1 Use the mouse to move the pointer to the left of the object.
- 2 Click the mouse button. A check mark (✓) appears to the left of the object to indicate that you have selected it.

or

- 1 Press *Esc* to close any open pull-down menu.
- 2 Press *Esc* to move the pointer from the main menu bar into the window.
- 3 Use the TAB key to move the pointer to the display subwindow.
- 4 Use the arrow keys to highlight the object.
- 5 Press the SPACE BAR. A check mark (✓) appears to the left of the object to indicate that you have selected it.

or

To select all items from an object list that supports checking multiple items:

- 1 Press accelerator button F8 and all of the items will be selected. To unselect all, press accelerator button F9.

Buttons

To select an object button or a command button, either:

- 1 Use the mouse to move the pointer to the button.
- 2 Click the mouse button. An action specified by the button occurs.

or

- 1 Press *Esc* to close any open pull-down menu.
- 2 Press *Esc* to move the highlight from the main menu bar into the window.
- 3 Use the TAB key to move the highlight to the button subwindow.
- 4 Use the arrow keys to highlight the button.
- 5 Press *Enter*. An action specified by the button occurs.

or

- 1 Use the TAB key to move the pointer to the button subwindow.
- 2 Type the first letter of the button you want to select.

Fields

Some dialog boxes display a one-line help message about a field when you select it. The message is displayed in the bottom border of the dialog box.

To select a field, either:

Use the arrow keys to highlight the field.

or

- 1 Use the mouse to move the pointer to the field.
- 2 Click the mouse button.

Many dialog boxes display an error message if you enter invalid data in a field. The message is displayed in the bottom border of the dialog box.

Some fields require you to enter values. To do so, select the field and type in a value. Use the BACKSPACE key to space to the left in the field; use the SPACE BAR to space to the right.

Some fields require you to select from a list of possible values. These fields are indicated with an up/down arrow.

Net Mgt Authenticate: ☐ No

Max Free Buffer Wait: ☐ 10 s

When you see an up/down arrow symbol, the current value is displayed in the field. To access the other options for this field, either move the pointer to the field and press the SPACE BAR, or click the mouse button. Continue cycling through the options until the desired value is displayed.

Some fields allow you to select from a menu of possible values, rather than cycling through the available options. These fields are indicated with a down arrow symbol.

Subnet 1: default_subnet Subnet 2:

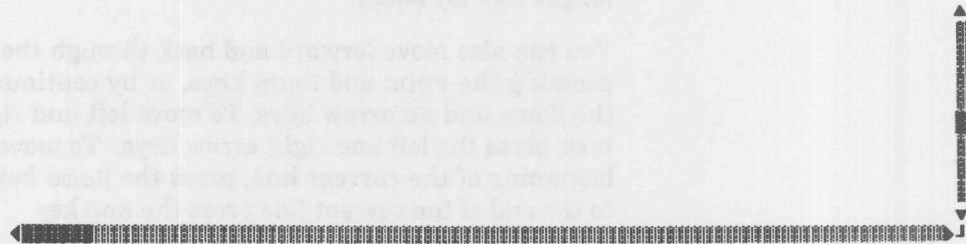
When you see a down arrow symbol, move the highlight to the arrow and press *Enter* or click the mouse button to open the menu. Select a value on the menu just as you would select a command on a pull-down menu from the main menu bar.

In some cases you may enter a value which is not listed as an option. For example, in the Node Spec window's Application Image Name field you may enter an App Image name that is not yet known to the system. Another example is the network variable list where NV arrays are displayed as one entry with the array size indicated in the array member index location. A dialog box prompts the user for the member index.

The last entry in this type of menu will always be "*BLANK*". Selecting this entry will blank out the field that is controlled by these menus. To close the menu without selecting a value, press *Esc*.

Scroll Bars

Some windows contain scroll bars, which allow you to move forward and backward or left and right through the text contained in a particular window.



When using the editor, you can:

Scroll forward through the text one line at a time:

To scroll forward one line at a time, click on the down arrow at the bottom of the scroll bar.

- 1 Use the mouse to move the pointer into the scroll bar just above the down arrow at the bottom of the scroll bar.
- 2 Click the mouse button.

To scroll backward one line at a time, click on the up arrow at the top of the scroll bar.

Scroll backward through the text one line at a time:

- 1 Use the mouse to move the pointer into the scroll bar just below the up arrow at the top of the scroll bar.
- 2 Click the mouse button.

To scroll right one window at a time, click on the right arrow on the scroll bar.

Scroll right through the text one window at a time:

- 1 Use the mouse to move the pointer into the scroll bar just to the left of the right arrow.
- 2 Click the mouse button.

To scroll left one window at a time, click on the left arrow on the scroll bar.

Scroll left through the text one window at a time:

- 1 Use the mouse to move the pointer into the scroll bar just to the right of the left arrow.
- 2 Click the mouse button.

Each time you click the mouse button new data appears in the window. Click as many times as you need to. When you reach the top, bottom, left, or right of the text clicking no longer has an effect.

You can also move forward and back through the text by pressing the PgDn and PgUp keys, or by continually pressing the down and up arrow keys. To move left and right through text, press the left and right arrow keys. To move to the beginning of the current line, press the Home key. To move to the end of the current line press the End key.

Using the Navigator

The Navigator home window, shown earlier in figure 3-1, is the top-level of a tree of windows that allow you to navigate through the object database. The Navigator windows are the interface to the LONBUILDER development tools. The Application Node window (figure 3-3) is a main branch of the Navigator tree. This window is a typical Navigator window:

- Object-type buttons. These are buttons used to select types of objects.
- An object list. This is a list of the selected type of object from the object database.
- Command buttons. These are buttons used to invoke operations on objects selected from the object list.

To use the Navigator:

- 1 Select an object-type button.
- 2 Select one or more objects from the object list.
- 3 Select a command button.

This guide provides explicit instructions on selecting Navigator object-type buttons, objects from the object list, and command buttons to accomplish LONBUILDER tasks .

To return to the Navigator home window at any time:

- 1 Press *Esc* once to close any open dialog box.
- 2 Press *Esc* once to close any open menu.
- 3 Press *F2* to open the Navigator home window (or select the Navigator command from the Window pull-down menu).

Figure 3-5 maps the windows that branch from the Navigator home window.

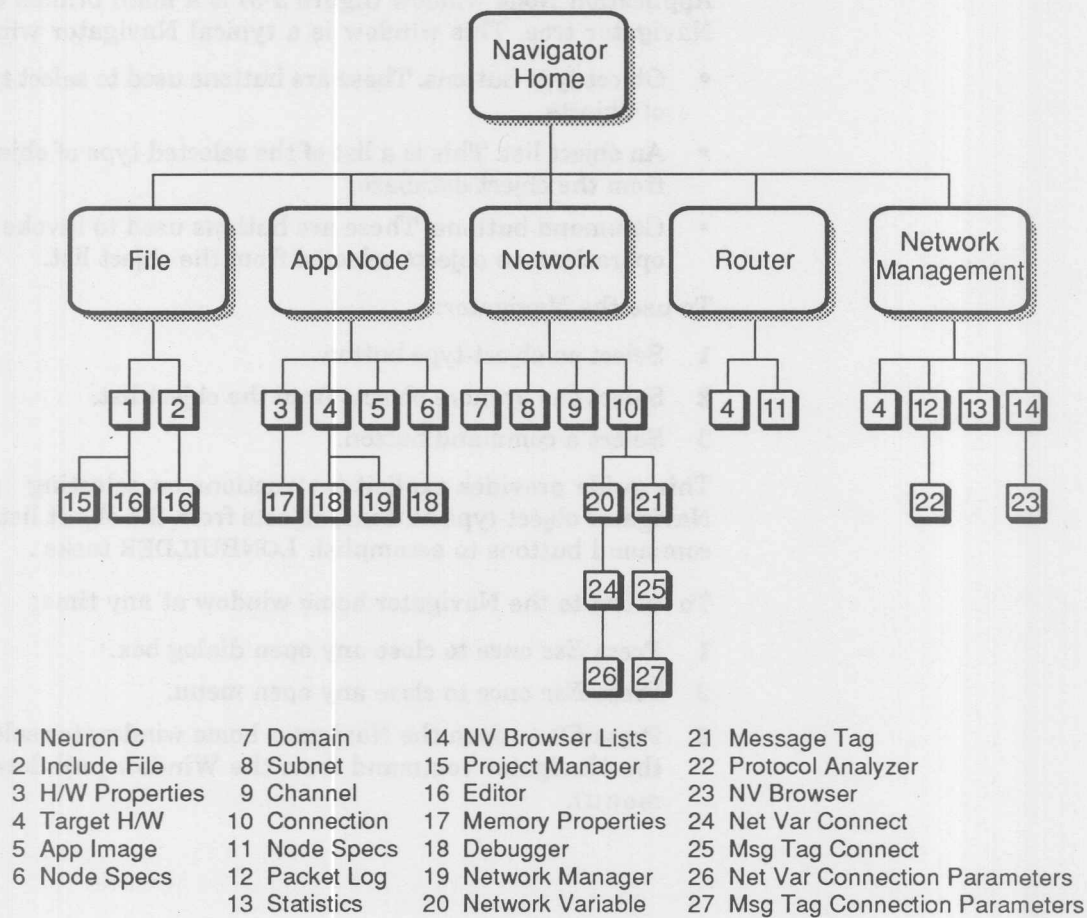


Figure 3-5. Navigating Through the LONBUILDER Object Database

Getting Help

The fields on many of the LONBUILDER application windows display a help message when you select them. Additionally, fields that accept new data display an error message if you enter invalid data in them.

General help on a variety of topics is available on the Help pull-down menu. To open and select from the Help menu, follow the instructions under *LONBUILDER Main Menu*, earlier.

The following general help topics on the LONBUILDER Editor are available when the Editor window is open:

Show Help	a brief description of the help system and how it works.
Help and Status	help on the Help system itself and on the <i>About</i> command, described under <i>System Menu</i> in Appendix A.
Cursor Movement	help on the basic cursor commands.
Quick Movement	help on the quick cursor movement commands.
Insert and Delete	help on the text insertion and deletion commands.
System	help on the commands that give access to system-level functions.
Files	help on the File pull-down menu commands.
Edit	help on the Edit pull-down menu commands.

Search and Replace	help on the Search pull-down menu commands.
View	help on the View pull-down menu commands.
Tabs	help on the Tab commands.
Options	help on the Options pull-down menu commands.
Project	help on the Project pull-down menu commands.
Macros	help on the Macro pull-down menu commands.
Windows	help on the Window pull-down menu commands.
Function keys	a quick reference list, showing all the commands assigned to function keys.

4

LONBuilder System and Project Configuration

This chapter describes how to:

- Create a directory structure for LONBuilder projects.
- Set the LONPROJ environment variable.
- Start a LONBuilder session.
- Configure LONBuilder system parameters.
- Configure LONBuilder project parameters.
- End a LONBuilder session.

Setting up Directories

A LONBUILDER application consists of application program files and database objects. You can use the same application programs in different network configurations. Each unique network configuration requires its own object database.

This section suggests how to create a directory structure for a LONBUILDER application, and for multiple applications that share program files. If you do not have elementary knowledge of the basic DOS commands, the DOS directory structure, and DOS pathnames, please refer to a beginner's guide to DOS before continuing with this section.

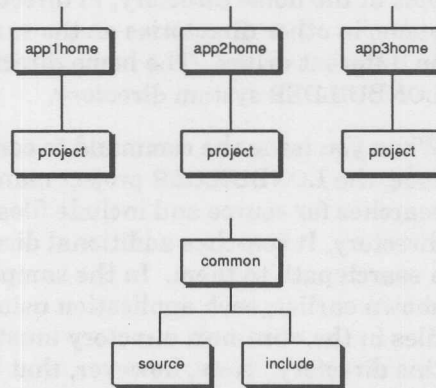
For a single LONBUILDER application, the recommended directory structure consists of a home directory and a project directory subordinate to the home directory. The figure below shows a sample directory structure for a single LONBUILDER application. You can use any valid DOS characters in your directory names.



For multiple applications, create the recommended directory structure for each application (you may create a common source file and include file directory if you have shared files). The next figure shows a sample directory structure for three applications.

Do not create home or project directories under the same directory you used to install the LONBUILDER Development Station software.

If multiple projects need to share common source and include files, it is possible to specify a search path that the compiler will use to retrieve these files. (See Chapter 5.)



Home Directory

The home directory is the top-level of an application's directory structure. You issue the command to start a LONBUILDER session, as described later in this chapter, from an application's home directory. This directory then becomes the current working directory. For example, if you start a LONBUILDER session from the **app3home** directory, shown in the sample directory structure earlier, **app3home** becomes the current working directory.

Application Directories

When you start a LONBUILDER session from the home directory, the home directory becomes the current working directory.

You can place an application's source and include files in as many DOS directories as you wish. The home directory is the directory that is current when you start the LONBUILDER software. You can place source and include files in the home directory, in directories subordinate to the home, in other directories on the same drive, or in directories on different drives. The home directory should not be in the LONBUILDER system directory.

When you issue the command to compile and link source code, the LONBUILDER project manager automatically searches for source and include files in the current working directory. It searches additional directories only if you specify a search path to them. In the sample directory structure shown earlier, each application using the source and include files in the **common** directory must specify a search path to this directory. Note, however, that the File button in the Navigator will not search the source or include search paths to retrieve files. See *Configuring LONBUILDER Project Parameters*, later, for instructions.

Project Directory

The project directory is where LONBUILDER places:

- Object database files created when you define the application nodes and their network configuration.
- Internal files generated when you build the application image and network image of nodes.

The database and internal files in the project directory are accessed by all LONBUILDER software tools, ensuring them a consistent view of your application.

The root directory may not be used as a home or project directory.

The home directory may not be used as a project directory.

You can place an application's project directory anywhere and name it any valid DOS name. In the samples, the project directories are subordinate to their applications' home directories. Each project directory has the same name: **project**. Placing the project directory subordinate to the home directory simplifies starting a LONBUILDER session as discussed in the following sections.

Setting the LONPROJ Environment Variable

When you set environment variables in the AUTOEXEC.BAT file, the variables do not take effect until you reboot DOS.

The LONPROJ environment variable specifies the relative or full pathname of the project directory. You can set the LONPROJ environment variable in the DOS AUTOEXEC.BAT file, or you can leave it unset.

In the sample directory structure shown earlier, all project directories are named **project**. To set the LONPROJ environment variable to the project directory of whichever application's home directory is current, use a relative pathname. Open the DOS AUTOEXEC.BAT file and enter

```
set LONPROJ=project
```

To set the LONPROJ environment variable to a specific project directory, use a full pathname. For example, to set LONPROJ to the **app3home** project directory on drive C:, open the DOS AUTOEXEC.BAT file and enter

```
set LONPROJ=C:\app3home\project
```

Save the new AUTOEXEC.BAT file and reboot the PC by pressing *Ctrl-Alt-Del* to invoke the new AUTOEXEC.BAT file.

Starting a LONBUILDER Session

The LONBUILDER system directory contains the LONBUILDER software, including the **lb** command. The system directory was automatically created during LONBUILDER software installation. During software installation, if you did not add the path to the LONBUILDER system directory to your AUTOEXEC.BAT file, you must do so before you issue the **lb** command.

To start a LONBUILDER session, change to the application's home directory and issue the **lb** command. The format of the command is:

```
lb [proj_dir_name] [/memory_option]
```

where *proj_dir_name* is the pathname of the project directory, and *memory_option* is the overlay memory type.

Specifying a project directory with the **lb** command is optional:

- If you start the LONBUILDER software with the command:

```
lb
```

the project directory defaults to the directory set by LONPROJ in the DOS AUTOEXEC.BAT file. (See *Setting the LONPROJ Environment Variable*, earlier, for details.)

- If you start the development system with the command:

```
lb proj_dir_name
```

the project directory is the directory specified by *proj_dir_name*.

You must give the project directory name with the **lb** command:

- When you haven't set the LONPROJ environment variable, or
- When you don't want to default to the project directory set by the LONPROJ environment variable.

The overlay memory type is selected as described under *Configuring LONBUILDER System Parameters*, later, in this chapter. The overlay memory type may also be selected when invoking LONBUILDER from the DOS command line with the following options:

/EXT - use extended memory for overlay management

/EXP - use expanded memory for overlay management

/DISK - read overlays directly from disk

After issuing the **lb** command to start a LONBUILDER session, you should see the following message:

LONBUILDER Developer's Workbench

Echelon, LON, and NEURON are registered trademarks of Echelon Corporation. LONTALK, LONBUILDER, LONWORKS, 3120 and 3150 are trademarks of Echelon Corporation.

Use of this software is subject to terms and conditions defined in the LONBUILDER Software License Agreement, the LONWORKS Development License Agreement, and the LONWORKS OEM License Agreement. The Neuron Chip Firmware described in the license agreements is contained in files with an SY prefix and NX extension (SY*.NX).

Copyright (c) 1991 by Echelon Corporation

Press any key or click mouse to continue

Configuring LONBUILDER System Parameters

The LONBUILDER software has four system options—color scheme, overlay memory, protocol analyzer buffer size, and I/O port start address—that are user-configurable. To configure these variables, follow these steps:

The overlay memory, buffer size, and I/O port start address selections made in this window do not take affect until you exit and restart the LONBUILDER software.

- 1 Open the Options pull-down menu.
- 2 Select the System command.



Figure 4-1 System Parameter Window

The type and amount of overlay memory can have a significant impact on LONBUILDER software performance. See the LONBUILDER Startup and Hardware Guide.

Overlay Memory

selects extended memory, expanded memory, or disk storage for LONBUILDER software overlays. The LONBUILDER software requires more than the 640 KB maximum memory available to DOS applications. To work around the DOS limit, the LONBUILDER software uses overlays that may be stored in extended memory, LIM 4.0 compatible expanded memory, or on disk.

Protocol
Analyzer Buffer
Size

The selection depends on how much cache and the type of extended or expanded memory, if any, is installed on your PC. An asterisk (*) marks the current setting. (The *LONBUILDER Startup and Hardware Guide* discusses the cache and overlay memory requirements for the LONBUILDER software.)

selects a buffer size from the available options: 30, 66, 114, or 255.

The protocol analyzer stores LONTALK packets in fixed sized buffers in a fixed sized space. More buffers are available when a smaller buffer size is selected, thereby increasing the performance of the protocol analyzer when logging is enabled. If LONTALK packets exceed the buffer size, the packet will be truncated by the protocol analyzer and all of the data will not be displayed.

If your application uses explicit messaging, set the buffer size to accept all the packets your application can generate. Otherwise, leave the size at 66 bytes.

Specifying a larger buffer size will reduce the number of buffers that the protocol analyzer has available, and thus increases the probability of a packet being missed. For the new buffer sizes to take effect, you must exit and then re-enter the LONBUILDER software. An asterisk (*) indicates the current setting.

Color Scheme

selects one of three different color schemes or monochrome for the LONBUILDER software menus and windows. The initial default is set to Color Set 1. To change the default, select one of the other color set choices. The color change goes into effect after selecting the dialog box's OK button. Select the preferred color, or select Monochrome if you have a monochrome screen (greyscale VGA).

Four-level displays, such as those found on some laptop computers, may look better if Color Set 1 is used.

I/O Port Start Address

sets the I/O port start address for accessing the LONBUILDER Interface Adaptor. The address must also be selected on the interface adaptor, as described in the *LONBUILDER Startup and Hardware Guide*. The interface adaptor occupies eight contiguous address locations. Start addresses range from 200 to 3F8 hex. Any address that does not conflict with your other I/O devices may be used. Suggested start addresses that usually do not cause conflicts are 270, 310, and 3A0 hex (The default is set to 310 hex.) Exit, then re-enter the LONBUILDER software for the new I/O port value to take effect.

Configuring LONBUILDER Project Parameters

This section describes how to configure project parameters:

- Build (compile, link, and load) options, which allow you to:
 - Specify when to stop compilations.
 - Invoke the editor when a compilation or build stops due to compile errors.
 - Generate information files about a build.
 - Stop the build process after compilation.
 - Load executables after creating network connections.
- Directories where the compiler and linker search for include files and NEURON C source files

To set build options, follow these steps:

- 1 Open the Options pull-down menu.
- 2 Select the Project command to open the Project Configuration window (see figure 4-2).

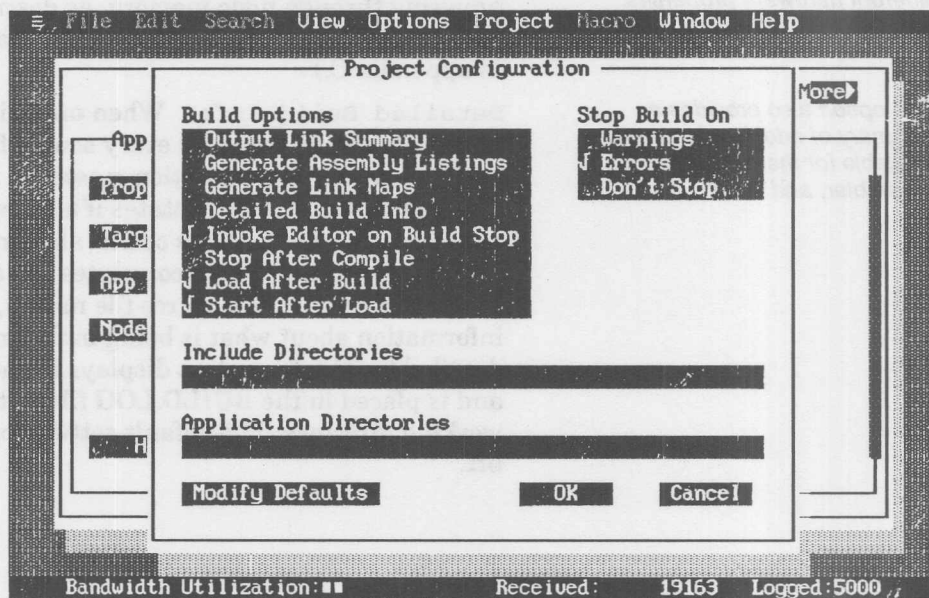


Figure 4-2. Project Configuration Window

- 3 Select the Build Options to set them on (check mark ✓) or off (blank) as desired. The Build Options and their definitions are:

Output Link Summary. When **on**, this option generates a link summary for the NEURON 3120 CHIP and the NEURON 3150 CHIP; when **off**, a link summary is not generated. The link summary displays in the Build window (shown in Chapter 7) and is placed in the BUILD.LOG file in the current working directory. The default setting for this option is **off**.

Generate Assembly Listings. When **on**, this option generates an assembler listing file, *filename.NL*, which is placed in the working directory; when **off**, an assembler listing file is not generated. The default setting for this option is **off**. Setting this option **on** results in slower builds and increased disk space usage.

Generate Link Map. When **on**, this option generates a detailed link map, *filename.MAP*, which is placed in the working directory; when **off**, a detailed link map is not generated. The default setting for this option is **off**. Setting this option **on** results in slower builds and increased disk space usage. (Link maps are useful for browsing through node memory, as described in Chapter 13. Portions of a sample detailed link map are annotated in appendix C.)

Detailed Build Info. When **on**, this option prints a recompilation message for every source file in the search path; when **off**, recompilation messages are not printed. A recompilation message states if a file will or will not be recompiled and why. This option also provides some useful information which correlates the application image names with the source file names, and other information about what is being built, and why. The detailed build information displays in the Build window and is placed in the BUILD.LOG file in the current working directory. The default setting for this option is **off**.

An assembly listing is generated for each NEURON C source file in an application image. There is a filename.NL for every NEURON C file compiled. The file name corresponds to the source file name.

There is one detailed link map generated per application image. The map name, filename.MAP, is constructed from the first four letters of the source file name plus a unique number to differentiate MAP files corresponding to different hardware properties.

This option also provides a summary of additional memory available for the compiler, assembler, and linker.

These three options (Stop After Compile, Load After Build, and Start After Load) affect the operation of the Automatic Build and Build All build commands. See Selecting a Project Build Command, in Chapter 7, for details.)

Invoke Editor on Build Stop. When **on**, this option invokes the program editor whenever the build process stops due to a compilation error; when **off**, the program editor is not invoked. The default setting for this option is **on**. (How you set the Stop Build On option determines when the build process stops. The Stop Build On option is also in the Project Configuration window and is described later in this section.

Stop After Compile. When **on**, this option stops the build process after compiling and before connecting; when **off**, the build process continues through connecting (i.e. binding). The default setting for this option is **off**.

Load After Build. When **on**, this option loads the executables after compiling and connecting; when **off**, the build process stops without loading. The default setting for this option is **on**.

Start After Load. When **on**, this option starts all the newly loaded target application nodes. The default setting for this option is **on**.

- 4 Select one of the Stop Build On options to set one of them **on**. The Stop Build On options and their definitions are:

Warnings

When **on**, this option stops compilations after warning messages. The default setting for this option is **off**.

Errors

When **on**, this option stops compilations after errors. The default setting for this option is **on**.

Don't Stop

When **on**, this option continues compilations regardless of warnings and errors. The default setting for this option is **off**.

■ Page 4-13 Addition

Add the following Build Option (follows Start After Load)

Use Extended Memory. When **on**, this option causes extended memory versions of the compiler, assembler, and linker to be used. The extended memory versions can handle larger NEURON C programs, but they are slower than the standard versions. The default setting for this option is **off**. To use this option, an extended memory manager must be installed which supports DPMI, VCPI, or LIM 4.0 compatible expanded memory emulation.

You can place your application source and include files in one or more DOS directories. When you invoke a LONBUILDER project build command, as described in Chapter 7, the system automatically searches the current working directory—the directory from which you started the LONBUILDER session—for source and include files. If source and include files are in other directories, you must tell the LONBUILDER IDE where to search for them.

5 Select the Application Directories field. Type the pathnames of the directories to be searched for .NC files. Separate each pathname with a semicolon (;). You don't need to specify the current working directory, because this directory is automatically searched first. You can use relative pathnames for directories that are relative to the current working directory; otherwise, use full pathnames. (Directories are searched in the order in which you list them. If multiple files of the same name are found in different directories, the first instance of the file is used.)

6 Select the Include Directories field. Type the pathnames of the directories to be searched for .H files. Separate each pathname with a semicolon (;). You don't need to specify the current working directory, because this directory is automatically searched first. You can use relative pathnames for directories that are relative to your current working directory; otherwise, use full pathnames. (Directories are searched in the order in which you list them. If multiple files of the same name are found in different directories, the first instance of the file is used.) The NEURON C and Include File buttons from the Navigator do not search the search path directories when displaying the list of files.

7 Select one of the following buttons:

The LONBUILDER system directory contains the LONBUILDER software. This directory was automatically created during LONBUILDER software installation.

Modify Defaults Saves to the LONBUILDER system directory. The project configuration parameters are applicable to every project that does not have its own specific project configuration.

OK Saves to the project directory. The project configuration parameters are applicable to the current project only.

Cancel Cancels without saving any changes made since the last save.

Using a DOS Shell

You may execute some simple DOS commands, without exiting LONBUILDER, by invoking a temporary DOS shell.

To invoke a temporary DOS shell, select `DOS Shell` from the `System` pull-down menu, (or type *Ctrl-Z* to access this same menu). The screen will clear and you will see the DOS prompt.

At this point, LONBUILDER is still in memory so you may not have enough memory to run other large programs. Also, you must not use any program that starts a TSR (resident) program, as this may interfere with the LONBUILDER software or cause the system to crash.

To return to your previous position in the LONBUILDER software, type `Exit` at the command prompt.

Ending a LONBUILDER Session

To end a LONBUILDER session, select `Exit` from the `System` pull-down menu or type *Alt-X*.

5

Editing Files

This chapter describes how to create and modify NEURON C source files using the LONBUILDER Editor. For instructions on programming with NEURON C, see the *NEURON C Programmer's Guide*.

Program Files

One of the tasks in the LONBUILDER development cycle is to define—create, modify, copy, and delete—the application's NEURON C source files and include files. This section describes how to perform these tasks using the LONBUILDER Editor.

Creating a Program File

You use the LONBUILDER Editor to create a NEURON C file or an include file. There are three ways to open the editor to create a new program file: (1) from DOS, (2) with the editor accelerator key, (3) from the Navigator window.

To open the editor from DOS, type the following:

```
LBE
```

The Editor window (see figure 5-3, later) opens with an UNTITLED file, which you can name later on.

To open the editor from DOS to create one to four files simultaneously, type the following command line:

```
LBE new1.nc [new2.nc] [new3.nc] [new4.nc]
```

The LONBUILDER Editor creates the files named in the command line and opens the Editor window (see figure 5-1, later) with a text window for each file.

To open the editor using its accelerator key, press *F3* from any LONBUILDER application window. The editor opens with the most recently edited file displayed in the text window. Open the **File** pull-down main menu and select the **Open** command, which prompts you for the name of the file you want to create.

To open the editor from the Navigator window, follow these steps:

- 1 Press F2 to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the File button to list the file object type selectors—NEURON C and Include File.
- 3 Select the NEURON C or Include file button to list the existing NEURON C or include file names in the object list.
- 4 Select the Create command button to open the Editor window (see figure 5-1, later) for an UNTITLED file, which you can name later on.

Modifying Program Files

You use the LONBUILDER Editor to modify a NEURON C file or an include file. There are three ways to open the editor to modify an existing program file: (1) from DOS, (2) with the editor accelerator, (3) from the Navigator window.

To open the editor from DOS to edit one to four files simultaneously, type the following command line:

```
LBE file1.nc [file2.nc] [file3.nc] [file4.nc]
```

The LONBUILDER Editor opens the existing files and creates any new files named in the command line. The Editor window (see figure 5-1, later) opens with a text window for each file.

On the LBE command line you can specify one to four existing files or you can specify a *file* mask to display a list of filenames from which to choose. For example, if you enter the following command line:

```
LBE *.NC
```

the editor displays a list of filenames matching the mask *.NC. You then pick the file to edit from a sorted directory listing. (Refer to the DOS manual that accompanies your PC for full details on how to specify DOS file masks. See *Directory Display*, later in this chapter, for directions on how to select a file from a directory listing.)

To open the editor using its accelerator key, press *F3* from any LONBUILDER application window. The editor opens with the most recently edited file displayed in the text window. Open the File pull-down main menu and select the Open command, which prompts you for the name of the file you want to edit.

To open the editor from the Navigator window, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the File button to list the file object type selectors—NEURON C and Include File.
- 3 Select the NEURON C or Include file button to list the existing NEURON C or include file names in the object list. These lists do not use the application file search paths specified in the Project/Options screen.
- 4 Select one of the filenames in the object list.
- 5 Select the Modify command button to open the Editor window (see figure 5-1, later) with a text window for the selected file.

Deleting Program Files

To delete a program file, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the File button to list the file object type selectors—NEURON C and Include File.
- 3 Select the NEURON C or Include file button to list the existing NEURON C or include file names in the object list.
- 4 Select one or more of the files.
- 5 Select the Delete command button to delete the selected files.

LONBUILDER Editor Window

The LONBUILDER Editor is a programming editor that is closely coupled with the LONBUILDER Integrated Development Environment (IDE). The Editor manages up to four text windows at once allowing you to:

- Choose multiple views of the same file, four different files, or mix and match.
- Move or copy text between windows.
- Zoom one window to fill the entire screen.
- Resize windows to get the right perspective.

The editor has a built-in macro system that makes repetitive editing tasks easy to accomplish. You can even invoke macros from within search operations. Figure 5-1 shows the Editor window for a new untitled file. Figure 5-2 shows the Editor window with two text windows and the top text window active. Figure 5-3 shows the Editor window with two text windows and the bottom text window active.

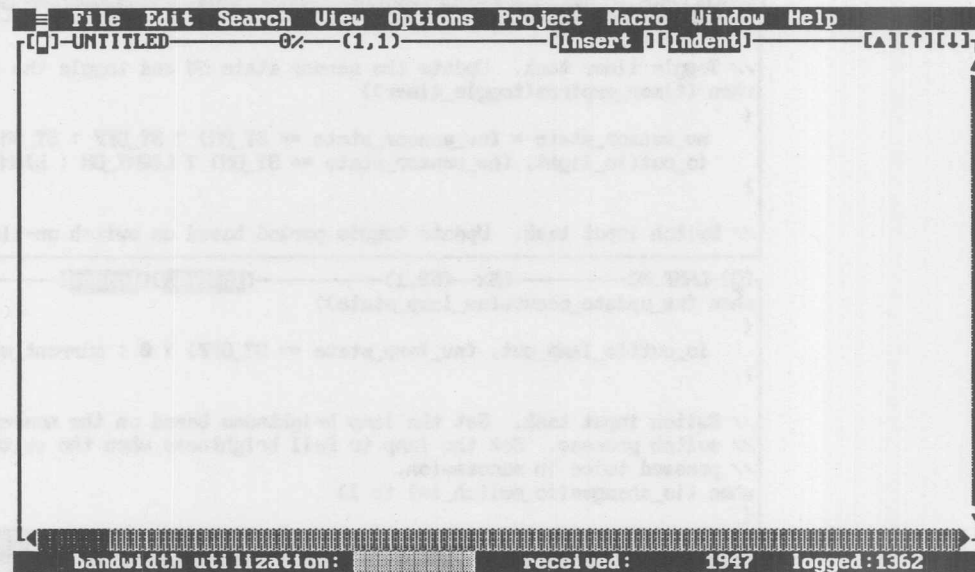


Figure 5-1. Editor Window for a New Untitled File

```

≡ File Edit Search View Options Project Macro Window Help
[ ]-SENSOR.NC 62%-(34,1) [Insert] [Indent] [A][I][U]
// Toggle timer task. Update the sensor state NU and toggle the light.
when (timer_expires(toggle_timer))
{
    nu_sensor_state = (nu_sensor_state == ST_ON) ? ST_OFF : ST_ON;
    io_out(io_light, (nu_sensor_state == ST_ON) ? LIGHT_ON : LIGHT_OFF);
}

// Switch input task. Update toggle period based on switch on-time.
LAMP.NC 65%-(53,1)
when (nu_update_occurs(nu_lamp_state))
{
    io_out(io_lamp_out, (nu_lamp_state == ST_OFF) ? 0 : current_width);
}

// Switch input task. Set the lamp brightness based on the number of
// switch presses. Set the lamp to full brightness when the switch is
// pressed twice in succession.
when (io_changes(io_switch_in) to 1)
{

```

Bandwidth Utilization: Received: 358 Logged: 0

Figure 5-2. Top Text Window Active

```

≡ File Edit Search View Options Project Macro Window Help
SENSOR.NC 62%-(34,1)
// Toggle timer task. Update the sensor state NU and toggle the light.
when (timer_expires(toggle_timer))
{
    nu_sensor_state = (nu_sensor_state == ST_ON) ? ST_OFF : ST_ON;
    io_out(io_light, (nu_sensor_state == ST_ON) ? LIGHT_ON : LIGHT_OFF);
}

// Switch input task. Update toggle period based on switch on-time.
[ ]-LAMP.NC 65%-(53,1) [Insert] [Indent] [A][I][U]
when (nu_update_occurs(nu_lamp_state))
{
    io_out(io_lamp_out, (nu_lamp_state == ST_OFF) ? 0 : current_width);
}

// Switch input task. Set the lamp brightness based on the number of
// switch presses. Set the lamp to full brightness when the switch is
// pressed twice in succession.
when (io_changes(io_switch_in) to 1)
{

```

Bandwidth Utilization: Received: 396 Logged: 0

Figure 5-3 Bottom Text Window Active

The top border in the LONBUILDER editor is always a status line, providing the following controls and information:

[■]	Closes the window when you click inside the brackets. Pressing <i>Ctrl-Minus</i> performs the same function.
FILENAME.EXT	Is the name and extension of the file being edited. A star symbol is displayed in front of the name if the file has not been written since the last change. Press <i>Ctrl-S</i> to write the file. Although the editor accepts complete path designations, for example: C:\DOCS\SAMPLE.DOC the drive and pathname are not displayed on the status line.
(RO)	Designates a read-only file.
xx%	Indicates the position of the cursor relative to the last character in the file.
(line,column)	Shows the line and column containing the cursor. The line is counted from the start of the file.
[Insert]	Indicates that Insert mode is in effect. If Insert mode is off, [Replace] is displayed instead. Click inside the brackets to change the mode, or press the <i>Ins</i> key.
[Indent]	Indicates that Autoindent mode is in effect. If Autoindent mode is off, [NoInd] is displayed instead. Click inside the brackets to change the mode, or press <i>Ctrl-F5</i> .
>R<	Indicates that macro recording is on. <i>Ctrl-M</i> toggles macro recording on and off; see <i>Macro Menu</i> in Appendix A for details.

>P<

Indicates that printing is in progress. This is displayed in the same location as the macro recording symbol.

[▲]

Zooms [▲] or unzooms [▼] the window when you click inside the brackets. Pressing *Ctrl-F2* performs the same function.

[↓]

Decreases the size of the window when you click inside the brackets. Pressing *Ctrl-F1* followed by pressing the down arrow key performs the same function.

[↑]

Increases the size of the window when you click inside the brackets. Pressing *Ctrl-F1* followed by pressing the up arrow key performs the same function.

If the ruler line has been turned on with the Toggle Ruler Line command (*Ctrl-F3*), the second line in the text window displays the locations of the current tab stops.

You can open up to four Editor windows at a time. Each window can contain text from a different file or text from the same file. You can divide Editor windows as long as they contain more than seven screen lines (counting the status and ruler lines).

You enter text in the editor in much the same way as you enter text on a typewriter, and most of the keys on the keyboard behave in the same fashion (press *Enter* to end each line, for example). The cursor always indicates where new text is entered.

The LONBUILDER Editor has menu and non-menu commands. The menu commands can be invoked by selecting a command from a pull-down menu on the main menu bar. The menu commands are described in Appendix A, *Main Menu Bar*. The non-menu commands are described in this section under *Basic Movement Commands* and *Basic Editing Commands*.

Basic Movement Commands

The basic movement commands move the cursor without altering text. In addition to the movement commands described in this section, movement commands appear under the *View Menu*, described in Appendix A. The first 2 basic movement commands use the mouse, the remainder use the keyboard.

Table 5-1. Basic Movement with the Mouse

<i>Command</i>	<i>Mouse</i>
Move cursor within window	Click mouse at new position
Scroll window	Click mouse within scroll bars

Table 5-2. Basic Movement with the Keyboard

<i>Command</i>	<i>Keystroke</i>
Beginning of Line	Home
Character Left	Left Arrow
Character Right	Right Arrow
End of Line	End
Equal Indent Down	Shift-Down
Equal Indent Up	Shift-Up
Line Down	Down Arrow
Line Up	Up Arrow
Match Delimiter	Ctrl-[
Page Down	PgDn
Page Up	PgUp
Scroll Window Down	Ctrl-Down
Scroll Window Up	Ctrl-Up
Tab Left	Shift-Tab
Tab Right	Tab
Word Left	Ctrl-Left
Word Right	Ctrl-Right

Move cursor within window (Click mouse at new position)

Clicking the mouse moves the cursor to the location of the mouse pointer. The mouse pointer must be within the current window. If the mouse pointer is inside a window that is not current, the first mouse click makes the window current and the second click moves the cursor.

Scroll window (Click mouse in scroll bars)

Clicking the mouse in a scroll bar scrolls the window based on which part of the scroll bar is selected. (See *Scroll Bars* in Chapter 3 for details.)

Beginning of line (Home)

Invoking *Beginning of line* moves the cursor to column 1 of the current line.

Character left (Left arrow)

Invoking *Character left* moves the cursor one character to the left. When the cursor reaches column 1, it stops.

Character right

(Right arrow)

Invoking *Character right* moves the cursor one character to the right. When the cursor reaches the right-hand edge of the text window, the text starts scrolling horizontally until it reaches the extreme right edge of the line (column 999), where it stops.

End of line

(End)

Invoking *End of line* moves the cursor to the end of the current line (the position following the last non-blank character on the line). Trailing blanks are always removed from all lines to conserve memory and disk space.

Equal indent down

(Shift-Down arrow)

Invoking *Equal indent down* moves the cursor to the beginning of the next line with the same indentation level as the current line. For example, if the first non-blank character in the current line is at column 20, the cursor is moved down to the next line that also begins at column 20.

Equal indent Up

(Shift-Up arrow)

Invoking *Equal indent up* moves the cursor to the beginning of the previous line with the same indentation level as the current line. For example, if the first non-blank character in the current line is at column 20, the cursor is moved up to the preceding line that also begins at column 20.

Line down

(Down arrow)

Invoking *Line down* moves the cursor to the line below. If the cursor is on the last line of the window, the window scrolls up one line.

Line up

(Up arrow)

Invoking *Line up* moves the cursor to the line above. If the cursor is on the top line of the window, the window scrolls down one line.

Match delimiter

(Ctrl-[)

Invoking *Match delimiter* moves the cursor to the delimiter that matches the delimiter currently under the cursor. If there is no match for the selected delimiter, the cursor does not move. You can match the following delimiters:

- braces: { and }
- angle brackets: < and >
- parentheses: (and)
- brackets: [and]
- comment markers: /* and */
- double quotes: "
- single quotes: '

The editor knows which direction to search for the match to a brace, angle bracket, parenthesis, and bracket. When searching for the match to a double or single quote, you must indicate if the cursor currently marks the beginning or ending quote. Use Ctrl-[to indicate the beginning quote; use Ctrl-] to indicate the ending quote. When searching for the match to a comment, indicate the beginning of the comment by placing the cursor over the slash; indicate the end of the comment by placing the cursor over the asterisk.

Page down

(PgDn)

Invoking *Page down* moves the cursor one page down with an overlap of one line.

Page up

(PgUp)

Invoking *Page up* moves the cursor one page up with an overlap of one line.

Scroll window down

(Ctrl-Down arrow)

Invoking *Scroll window down* scrolls down toward the end of the file, one line at a time (the entire window scrolls up). The cursor remains on its line until it reaches the top of the window.

Scroll window up

(Ctrl-Up arrow)

Invoking *Scroll window up* scrolls up toward the beginning of the file, one line at a time (the entire window scrolls down). The cursor remains on its line until it reaches the bottom of the window.

Tab left

(Shift-Tab)

Invoking *Tab left* moves the cursor to the previous tab stop, if table tabs are not in effect. Unlike the *Tab right* command described below, this command moves only the cursor. It never moves text, whether in Insert mode or Replace mode. If table tabs are in effect, this command does nothing.

Tab right

(Tab)

Invoking *Tab right* moves the cursor to the next tab stop. In Insert mode, any text to the right of the cursor is moved along with it; in Replace mode, only the cursor is moved.

Word left

(Ctrl-Left arrow)

Invoking *Word left* moves the cursor to the beginning of the word to the left, using the NEURON C definition of a word. This command works across line breaks.

Word right

(Ctrl-Right arrow)

Invoking *Word right* moves the cursor to the beginning of the word to the right, using the NEURON C definition of a word. This command works across line breaks.

Basic Editing Commands

The basic editing commands alter text. In addition to the editing commands described in this section, editing commands appear under the various pull-down menus on the main menu bar, described in Appendix A. The first four basic editing commands use the mouse, the remainder use the keyboard.

Table 5-3. Basic Editing with the Mouse

<i>Command</i>	<i>Mouse</i>
Select Block	Click and drag the mouse
Select Column	Shift + click and drag the mouse
Select Line	Click mouse on left border
Select Word	Double click mouse on a word

Table 5-4. Basic Editing with the Keyboard

Command	Keystroke
Change to Lower Case	Ctrl-F6
Change to Upper Case	Ctrl-F7
Copy	Ctrl-Ins, Ctrl-C
Copy Append	Shift +Ctrl-C
Cut	Shift-Del, Ctrl-X
Cut Append	Shift + Ctrl-X
Delete Character Left	Backspace
Delete Character Right	Del
Delete Line	Ctrl-D
Delete to End of Line	Ctrl-K
Delete Word Left	Ctrl-Backspace
Delete Word Right	Ctrl-Del
Insert Control Character	Ctrl-Shift-Ins
Insert New Line	Enter
Open New Line	Ctrl-Enter
Paste	Shift-Ins, Ctrl-V
Select Line	Ctrl-L
Select Word	Ctrl-Shift-Right
Shift Block Left	Shift-Left
Shift Block Right	Shift-Right

Select block (Click and drag the mouse)

Clicking and dragging the mouse over text both highlights and selects a block of text. The character underneath the mouse pointer is not included in the block.

Select column (Shift plus click and drag the mouse)

Pressing *Shift* while clicking and dragging the mouse down a column of text selects the column.

Select line**(Click on left border)**

Clicking within the left border of a text window selects the line to the right of the mouse pointer. Multiple lines may be selected by dragging the mouse pointer along the left border.

Select word**(Double click on word)**

Double-clicking within a text window selects the word under the mouse pointer.

Change to lowercase**(Ctrl-F6)**

Invoking *Change to lowercase* changes the characters in the current block to their lowercase equivalent. If no block is selected, the character at the cursor is changed.

Change to uppercase**(Ctrl-F7)**

Invoking *Change to uppercase* changes the characters in the current block to their uppercase equivalent. If no block is selected, the character at the cursor is changed.

Copy**(Ctrl-Ins, Ctrl-C)**

Invoking *Copy* copies the marked and displayed block to the clipboard. The original block is left unchanged.

Copy append

(Shift + Ctrl-C)

Invoking *Copy append* copies the marked and displayed text and appends it to the text currently in the clipboard.

Cut

(Shift-Del, Ctrl-X)

Invoking *Cut* cuts the marked and displayed block to the clipboard and deletes the original.

Cut append

(Shift + Ctrl-X)

Invoking *Cut append* cuts the marked and displayed text and appends it to the text currently in the clipboard.

Delete character left

(Backspace)

Invoking *Delete character left* moves the cursor one character to the left and deletes the character positioned there. Any characters to the right of the cursor are moved one position to the left. If the cursor is at column 1 at the time the command is given, the current line and the previous line are joined at the cursor.

Delete character right

(Del)

Invoking *Delete character right* deletes the character at the cursor and moves any characters to the right of the cursor one position to the left. If the cursor is at the end of the line, the current line and the next line are joined at the cursor. **If a block is highlighted, the entire block is deleted.**

Delete line

(Ctrl-D)

Invoking *Delete line* deletes the line containing the cursor and moves any lines below it up one line. The cursor moves to column 1 of the next line.

Delete word left

(Ctrl-Backspace)

Invoking *Delete word left* deletes the word to the left of the cursor.

Delete word right

(Ctrl-Del)

Invoking *Delete word right* deletes the word to the right of the cursor.

Delete to end of line

(Ctrl-K)

Invoking *Delete to end of line* deletes all text from the position of the cursor to the end of the line.

Insert control character

(Ctrl-Shift-Ins)

Invoking *Insert control character* allows a control character to be entered into the text. For example, pressing *Ctrl-Shift-Ins* and then *Ctrl-B* would insert a Ctrl-B into the text. Control characters are displayed as highlighted uppercase letters.

Insert new line

(Enter)

In Insert mode, invoking *Insert new line* inserts a line break at the cursor's position. If Autoindent mode is in effect, the cursor moves to the next line and to the same column as the first non-blank character in the previous line; otherwise, it moves to column 1 of the new line.

In Replace mode, invoking *Insert new line* moves the cursor to column 1 of the next line without inserting a new line, whether Autoindent mode is in effect or not. A new line is inserted if you're in Replace mode and the cursor is on the last line of the file.

Open new line

(Ctrl-Enter)

Invoking *Open new line* inserts a line break at the cursor's position. The cursor does not move.

Paste

(Shift-Ins, Ctrl-V)

Invoking *Paste* inserts the clipboard at the current cursor position. The clipboard is not changed

Select line

(Ctrl-L)

Invoking *Select line* selects the entire line at the cursor's position.

Select word

(Ctrl-Shift-Right)

Invoking *Select word* selects a single word. If the cursor is positioned within a word, that word is selected. If the cursor is not within a word, then the word to the right of the cursor is selected. And if there is no word to the right of the cursor, then the word to the left is selected.

Shift block left

(Shift-Left)

Invoking *Shift block left* moves all selected lines one column to the left. The characters in column 1 are deleted. Column 1 must be included in the block.

Shift block right

(Shift-Right)

Invoking *Shift block right* moves all selected lines one column to the right. A blank is inserted at column 1. Column 1 must be included in the block.

Prompt Editor

When it's necessary for you to enter a number, a filename, or any other information beyond a single keystroke, the LONBUILDER Editor displays a dialog box: a pop-up window where you enter your response (the prompt itself is usually displayed in the border of the dialog box). Generally, a default answer is displayed, which you can accept by pressing *Enter*. You can also cancel the operation by pressing *Esc*. To enter a new response, type as your first character either a text character, *Ctrl-Shift-Ins*, *Ctrl-D*, or *Ctrl-V*.

When you are entering text in a dialog box, you are using the *prompt editor*, a special line editor with its own set of commands. The special command keys are listed in Table 5-5, below.

Table 5-5. Prompt Editor Commands

Command	Keystroke
Accept Entry	Enter
Beginning of Line	Home
Cancel	Esc
Character Left	Left Arrow
Character Right	Right Arrow
Delete Character Left	Backspace
Delete Character Right	Del
Delete Line	Ctrl-D
Delete to End of Line	Ctrl-K
End of Line	End
Help	F1
Insert Control Character	Ctrl-Shift-Ins
Paste	Ctrl-V
Restore Line	Ctrl-R
Toggle Insert Mode	Ins
Word Left	Ctrl-Left
Word Right	Ctrl-Right

Most of these commands work the same way as their equivalents in the main editor, described earlier under *Basic Editing Commands*. The exceptions are *Accept entry*, *Cancel*, and *Restore line*. In the editor proper, these commands either don't exist (*Accept entry* and *Cancel*) or are only available in the menu bar (*Restore line*). The *Restore Line* command restores the response to its initial contents.

Directory Display

Several commands prompt for a filename. To display the contents of a directory when prompted for a filename, enter a file mask at the dialog box; for example: C:\LONAPP*.NC shows you all the files in sub-directory LONAPP on drive C: that have the extension .NC. If a drive or path is not specified, the default directory is assumed.

Once a mask is entered, all files matching the search criteria are displayed in sorted order in a dialog box. The first filename in the window is highlighted with a selection bar. If the window is too small for all the filenames, use the scroll bar to scroll through the list, or scroll through the list using the keys:

<i>PgUp</i> and <i>PgDn</i>	scrolls the display one window. The first filename in the window is highlighted with a selection bar.
<i>Ctrl-PgUp</i>	scrolls to the first filename in the list, which is highlighted with a selection bar.
<i>Ctrl-PgDn</i>	scrolls to the last filename in the list, which is highlighted with a selection bar.

To pick a filename from a directory listing, use the arrow keys to move the selection bar to the filename and press *Enter*, or move the mouse pointer to the filename and double click. You can also pick a filename by entering one or more characters to move the selection bar to the first name that starts with the entered characters; if none of the names start with the entered characters, the selection bar moves to the closest match.

Using the Build Log with the Editor

If the Invoke Editor on Build Stop project parameter is set **on** (as described under *Configuring LONBUILDER Project Parameters* in Chapter 4), whenever a build procedure stops during compilation, the LONBUILDER software invokes the editor as described under *Selecting a Project Build Command* in Chapter 7. The Editor window opens with the BUILD.LOG file displayed in one window and the source file containing the first compile error in a text window. The first compile error in the BUILD.LOG file is displayed and highlighted in the BUILD.LOG window; in the text window, the line of code that caused the error is highlighted and, for color sets, colored to indicate the seriousness of the error:

Color (for color sets)	Type of Error
Yellow	warnings
Red	serious or fatal error

Figure 5-4 shows the Editor window with the BUILD.LOG window and a program file window open.

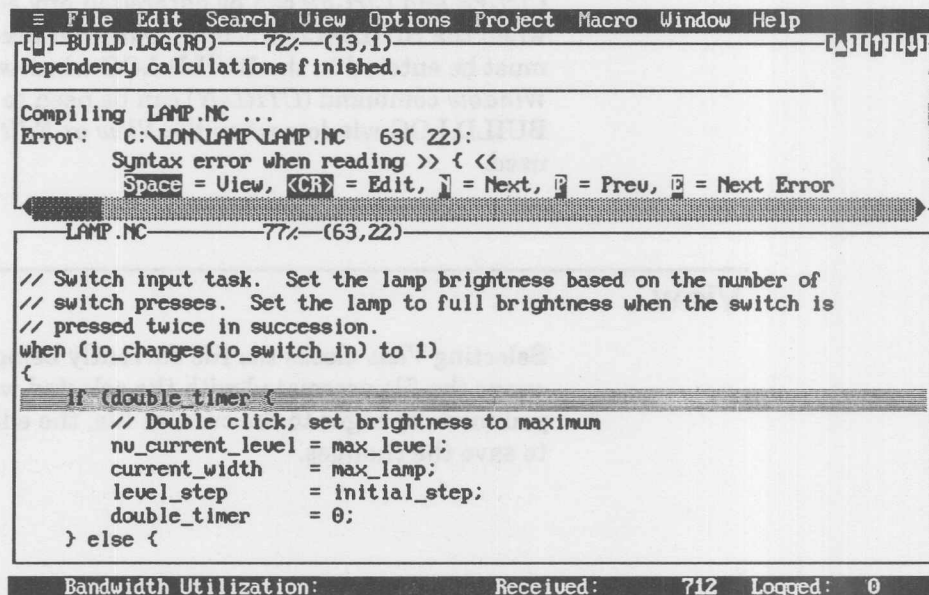


Figure 5-4. Editor Window with BUILD.LOG Window and Text Window

You use the commands listed in table 5-6 to move through the BUILD.LOG window and highlight compile errors. As each compile error in the BUILD.LOG is highlighted, the line of code that caused the error is highlighted in the text window. The editor opens and closes program files as needed when you invoke either the *View* or *Edit* command.

Table 5-6 lists the five command buttons and their accelerator keys found on the BUILD.LOG Editor window.

Table 5-6. BUILD.LOG Editor Window Commands

<i>Command</i>	<i>Accelerator Key</i>
View	Space
Edit	<CR>
Next	N or Ctrl-F8
Previous	P
Next Error	E or Ctrl-F9

Ctrl-F8 and *Ctrl-F9* can be entered in any editor window when the BUILD.LOG is open. All other accelerator keys must be entered in the BUILD.LOG window. The *Next Window* command (*CTRL-N*) can be used to return to the BUILD.LOG window when the *View* or *Edit* commands are used.

View

(Space)

Selecting *View* closes the file currently being edited, and opens the file associated with the selected warning or error. If you made changes to the current file, the editor prompts you to save the changes.

Edit**(<CR>)**

Selecting *Edit* closes the file currently being edited, opens the file associated with the selected warning or error, and moves the cursor to the line that caused the error in the newly opened file. If you made changes to the current file, the editor prompts you to save the changes.

Next**(N or Ctrl-F8)**

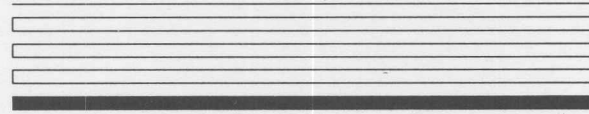
Selecting *Next* finds the next warning or error in the BUILD.LOG window.

Previous**(P)**

Selecting *Previous* finds the previous warning or error in the BUILD.LOG window.

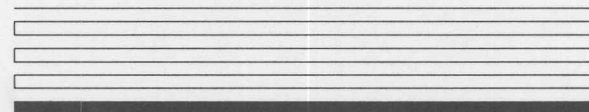
Next Error**(E or Ctrl-F9)**

Selecting *Next Error* finds the next error in the BUILD.LOG window.



Part 2

Node Implementation, Debug, and Test



6

Defining and Installing Application Nodes

This chapter describes how to define and install application nodes. Application nodes may be NEURON Emulators, Single Board Computers (SBCs), or custom nodes based on user-developed target hardware. In addition, this chapter describes how to view, rename, and import network variable descriptions.

Application Node Definition

An application node consists of target hardware with a system image, an application image, and a network image loaded in its memory. The system image contains the NEURON CHIP firmware. The system image must be loaded in ROM for a NEURON 3150 CHIP, and is on-chip for the NEURON 3120 CHIP. The system, application, and network image may be loaded into an application node's memory over the LONBUILDER backplane (for backplane resident emulators and SBCs), or may be pre-loaded into a custom node. The application and network images may also be loaded over the network by the network manager for remote nodes. A node may also update its own network image using the NEURON CHIP self installation functions (*see the NEURON CHIP-based Installation engineering bulletin.*)

A node is defined within the LONBUILDER software with a *node specification* stored in the object database. The node specification identifies the target hardware, application image, and network parameters for a node.

Node specifications are created from the Navigator using the App Node object type. To create node specifications, follow these steps:

- 1 Press F2 to open the Navigator home window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs (see figure 6-1).

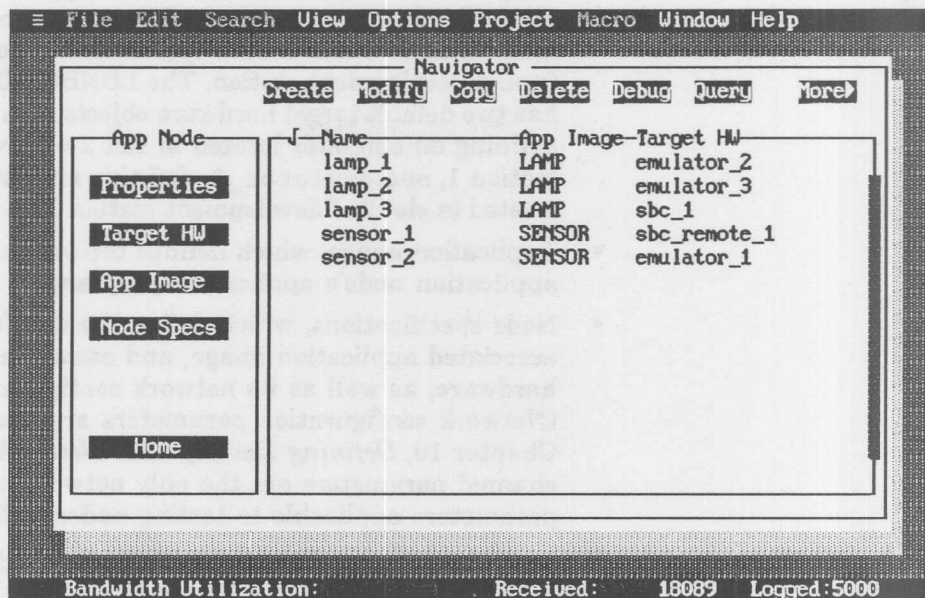


Figure 6-1. Application Node Selector Window

The Properties, Target HW, App Image, and Node Specs buttons are used to create and modify the following object definitions in the object database:

- Hardware properties, which define the NEURON CHIP properties of the application node's target hardware. The LONBUILDER software has default property parameters, default_hw_props, defining a NEURON 3150 CHIP with a 10 MHz input clock rate and a default external memory map consisting of 32 Kbytes of ROM.

- Target hardware, which defines the application node's target hardware—emulator, SBC, or custom node—and its location in either a development station or remote from a development station. The LONBUILDER software has two default target hardware objects, `emulator_1`, defining an emulator located in slot 2 of development station 1, and `emulator_2`, defining an emulator located in slot 3 of development station 1.
- Application image, which defines the origin of the application node's application program.
- Node specifications, which define the node's name, associated application image, and associated target hardware, as well as its network configuration. (Network configuration parameters are described in Chapter 10, *Defining Development Networks*. The channel parameters are the only network configuration parameters applicable to testing nodes individually.)

Every application node is physically configured for a communications medium or channel, and has channel parameters that define that configuration. The LONBUILDER system has default channel parameters, `default_channel`, which defines a direct connection with a backplane transceiver to the development station backplane at a 1.25 Mbps bit rate. These defaults are appropriate for the initial testing and debugging of most application nodes. If you want to test nodes on a different medium or at a different rate from the defaults, go to *Defining Channels* in Chapter 10 for instructions. Then, return to this chapter.

The default properties and target hardware objects are also appropriate for the initial testing and debugging of most application nodes. If they are appropriate for your nodes, go to *Defining Node Specifications*, later in this chapter.

If the defaults are not appropriate for your initial testing on emulators or if you're ready to test on SBCs and custom nodes, you must define new property and target hardware objects. To do this, you can copy the default object definitions and change them as needed. You must define the hardware properties and channels before the target hardware objects.

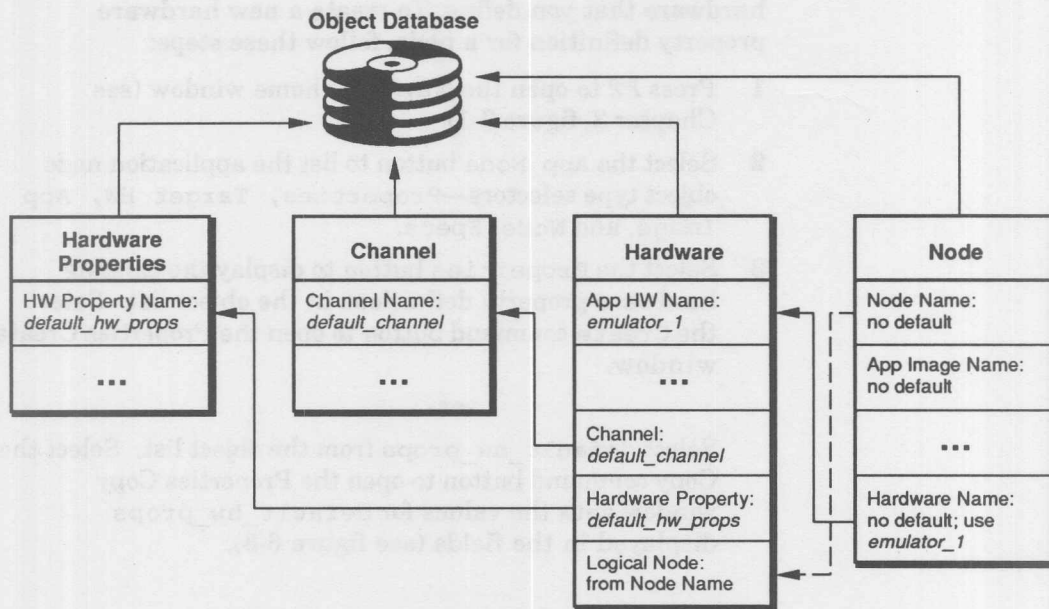


Figure 6-2. Relationship of Default Hardware Definitions

Defining Properties

Every application node must have the properties of its target hardware defined in the object database. Two or more application nodes can have the same properties definition. The object database is created with default hardware properties, `default_hw_props`. The defaults in `default_hw_props` are listed in this section.

If no other hardware property definitions exist, the default properties are automatically assigned to any target hardware that you define. To create a new hardware property definition for a node, follow these steps:

- 1 Press **F2** to open the Navigator home window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the Properties button to display the existing hardware property definitions in the object list. Select the Create command button to open the Properties Create window.

-or-

Select `default_hw_props` from the object list. Select the Copy command button to open the Properties Copy window with the values for `default_hw_props` displayed in the fields (see figure 6-3).

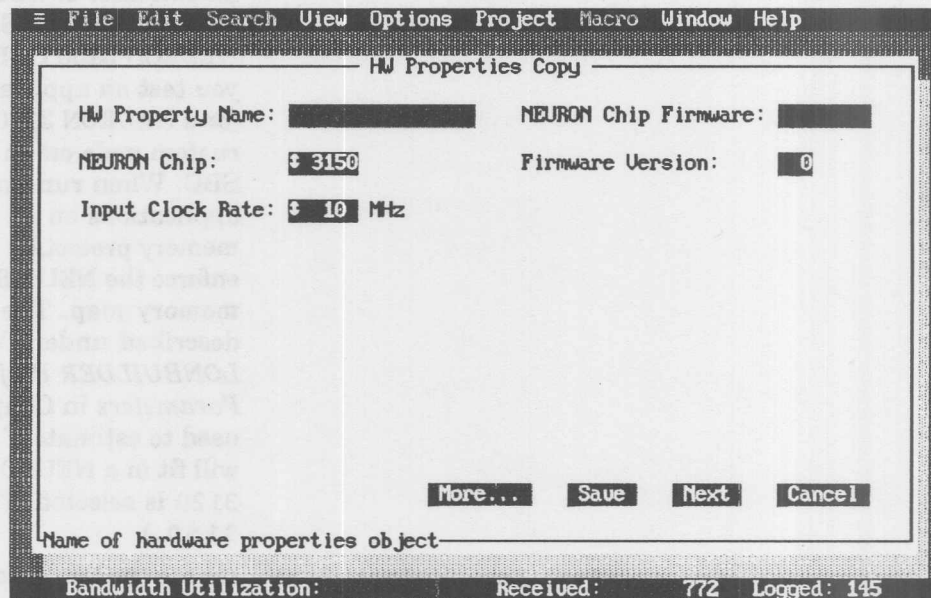


Figure 6-3. Hardware Properties Copy Window

- 4 Select fields and enter new values as needed. The fields and their defaults are:

HW Property Name	enter a 1- to 16-character unique name for the hardware property definition. You can use letters, numbers, and underscores in the name. No spaces are allowed.
------------------	--

When running 3120 applications on the emulator or SBC, the `clockedge +/-` option is not available for triac output (`clockedge +` and `clockedge -` are available).

NEURON CHIP

choose the model number of the NEURON CHIP: 3150 or 3120. You can choose 3120 for application nodes being tested on an emulator or SBC, even though the emulator and SBC use a NEURON 3150 CHIP. This lets you test an application defined for a NEURON 3120 CHIP-based custom node on an emulator or SBC. When running 3120 applications on an emulator, the memory protection is set up to enforce the NEURON 3120 CHIP memory map. The link map, described under *Configuring LONBUILDER Project Parameters* in Chapter 4, can be used to estimate if an application will fit in a NEURON 3120 CHIP if 3120 is selected. (The default is 3150.)

Input Clock Rate

choose the input clock rate of the NEURON CHIP. Possible rates are 10 MHz, 5 MHz, 2.5 MHz, 1.25 MHz, .625 MHz. (The default is 10 MHz.)

NEURON Chip Firmware

enter the name of the system image file. Normally, this field should be left blank, and a default system image will automatically be chosen based on the type of target hardware and NEURON CHIP. (The default is **blank**.)

Firmware Version

enter the version number (2 - 255) of the system image files you wish to use. If you enter a 0, the system will choose the default. For additional information on how this default is selected, refer to the discussion in Chapter 7 on *Using Multiple Firmware Versions*. (The default is 0)

5 If you have selected 3150 for the NEURON CHIP model, define the memory properties as described in the next section.

6 Select buttons as follows:

SAVE

creates a new hardware property definition with the values in the fields. After saving, enter new values to create another hardware property definition, or select CANCEL to return to the Navigator window.

NEXT

displays the values of the next hardware properties object selected.

CANCEL

returns you to the Navigator window without creating a new definition.

Defining Memory Properties

The Memory Properties screen allows you to configure four types of off-chip memory: ROM, EEPROM, RAM, and I/O.

The off-chip memory parameters only apply to the NEURON 3150 CHIP. For this NEURON CHIP, memory is allocated as depicted in figure 6-4. If you have selected 3120 for the NEURON CHIP model, the memory is allocated as shown in figure 6-5.

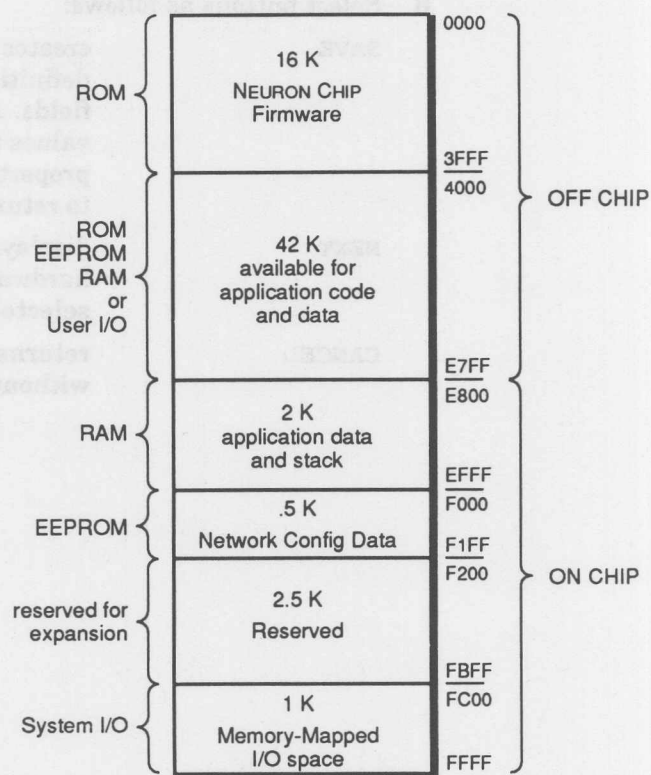


Figure 6-4. NEURON 3150 CHIP Memory Allocation

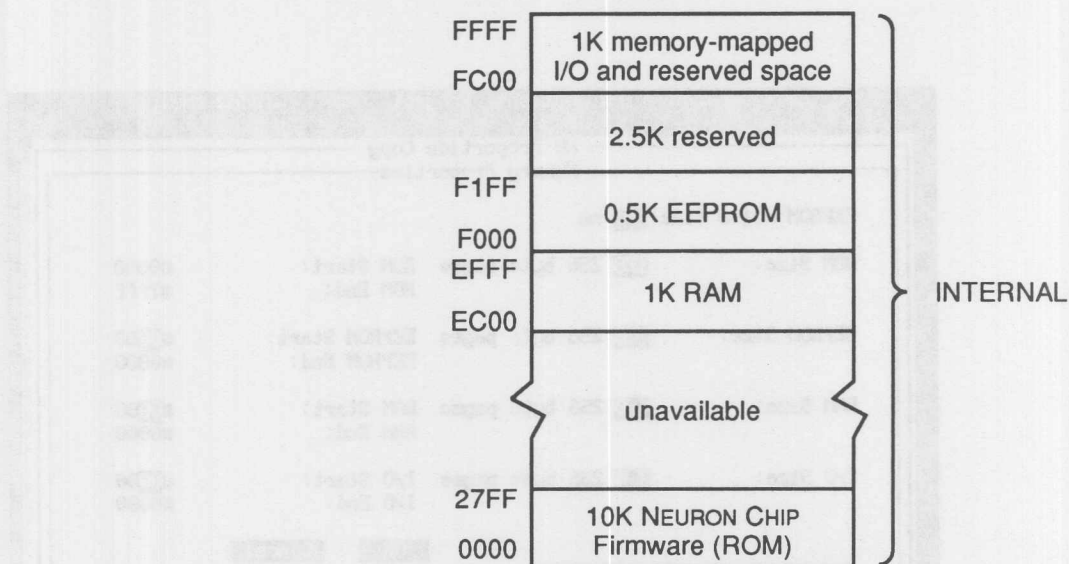


Figure 6-5. NEURON 3120 CHIP Memory Allocation

The 58 Kbytes of available off-chip memory is divided into 256 byte pages. A node's hardware properties contain its off-chip memory configuration. You can configure off-chip memory size in contiguous 256-byte pages for off-chip ROM, EEPROM, RAM, and I/O space. The total size of the ROM, EEPROM, RAM, and I/O space cannot exceed 58 Kbytes.

If you have selected 3150 for the NEURON CHIP model, follow these steps.

- 1 While in the HW Properties screen, if you have chosen 3150 for the NEURON CHIP model, select the More button.

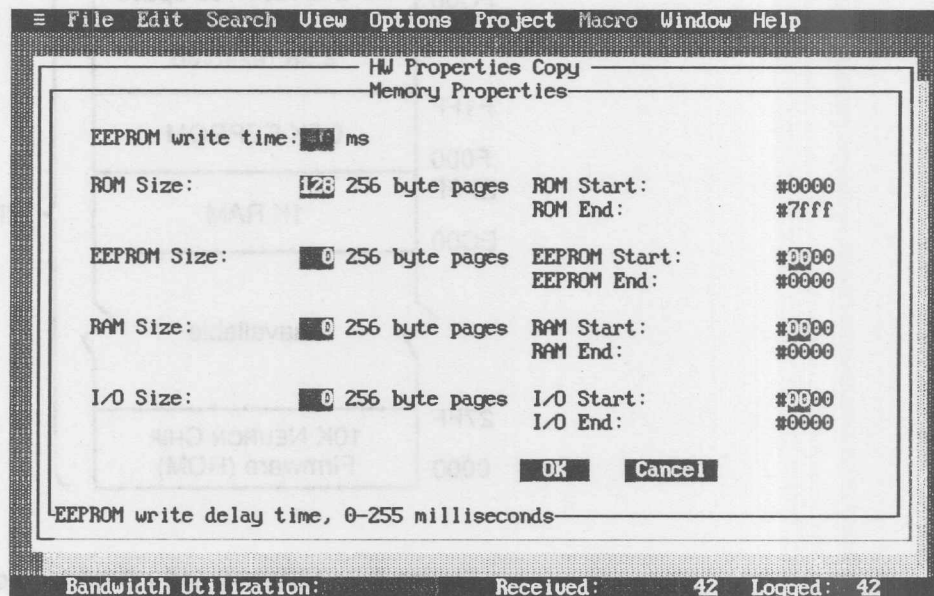


Figure 6-6. Memory Properties Window

- 2 Enter the values as required in the Memory Properties window. The address ranges for ROM, EEPROM, and RAM must increase in this order. The address range for the I/O space may be placed in any order. None of these areas may overlap or wrap.

Note that EEPROM in this context refers to non-volatile read/write memory. If you wish to use a non-volatile or battery-backed RAM, define it as EEPROM, with a write delay of 0 ms in the Memory Properties window.

EEPROM Write Time	enter the off-chip memory EEPROM write time. Valid values range from 0 to 255 ms. (The default is 10 ms . The default is suitable for most EEPROM chips. If you are using nonvolatile RAM in place of off-chip EEPROM, set this field to 0.)
ROM Size	enter the integer value representing the size in 256-byte pages of off-chip ROM. The first 16 Kbytes of ROM is used for the NEURON CHIP firmware. Valid values range from 64 to 232 (16 Kbytes to 58 Kbytes). (The default is 128 (32 Kbytes).) If an application program is to reside in on-chip EEPROM memory only, set this field to 64 (16 Kbytes), and leave EEPROM Size set at 0.
ROM Start	is a read-only field. ROM always starts at 0.
ROM End	is a read-only field. It changes automatically as the value for ROM Size changes.
EEPROM Size	enter the integer value representing the size in 256-byte pages of off-chip EEPROM. Valid values range from 0 to 168 (42 Kbytes). (The default is 0.)

EEPROM Start	enter the EEPROM start address or 0, meaning no off-chip EEPROM. Valid start addresses range from 4000 hex to E700 hex. The address you enter must be greater than the ROM End address. (The default is 0.)
EEPROM End	is a read-only field. It changes automatically as the values for EEPROM Size and EEPROM Start change.
RAM Size	enter the integer value representing the size in 256-byte pages of off-chip RAM. Valid values range from 0 to 168 (42 Kbytes). (The default is 0.)
RAM Start	enter the RAM start address or 0, meaning no off-chip RAM. Valid start addresses range from 4000 hex to E700 hex. The address you enter must be greater than the EEPROM End and ROM End addresses. (The default is 0.)
RAM End	is a read-only field. It changes automatically as the values for RAM Size and RAM Start change.

If the application running on a node with memory mapped I/O will be protected from network management reads and writes, and you want the I/O space similarly protected, start the I/O area immediately following the ROM area, before any other memory areas.

The debugger cannot access memory-mapped I/O devices. Using the debugger to access memory locations in the I/O space will produce incorrect results.

I/O Size

enter the integer value representing the size in 256-byte pages of memory-mapped I/O space. This space will be reserved for application use, and will not be read or write protected when the application is running on an emulator. See the emulator documentation in the *LONBUILDER Startup and Hardware Guide* for instructions on how to use memory-mapped I/O. Valid values range from 0 to 168 (42 Kbytes). (The default is 0.)

I/O Start

enter the memory-mapped I/O start address or 0 (meaning no memory mapped I/O space). Valid addresses range from 4000 hex to E700 hex. The address you enter must be greater than the ROM End address. (The default is 0.)

I/O End

is a read-only field. It changes automatically as the values for I/O Size and I/O Start change.

3 Select buttons as follows:

OK

returns you to the Hardware Properties window. The changes you have made in the Memory Properties window are retained. You must still select SAVE in the Hardware Properties window to save these definitions.

CANCEL

returns you to the Hardware Properties window without retaining any changes you have made in the Memory Properties window.

Defining Target Hardware

Every application node must have its target hardware defined in the object database. Application programs are typically debugged on emulator target hardware. Once debugged, application programs may be tested on remote SBC target hardware. After debugging on an emulator and testing on a remote SBC, application programs can be loaded on custom node target hardware.

The object database is created with two default target hardware objects: `emulator_1` and `emulator_2`. The defaults for `emulator_1` and `emulator_2` are listed in this section.

To create a new target hardware object for a node, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the Target HW button to display the existing target hardware definitions in the object list.
- 4 Select `emulator_1` or `emulator_2` from the object list. Select the Copy command button to open the Target HW Copy window with the `emulator_1` or `emulator_2` definition displayed (see figure 6-7).

-or-

Select the Create command button to open the Target HW Create window.

File Edit Search View Options Project Macro Window Help

App HW Copy

App HW Name:

HW Type: Location:

Channel Name: Priority:

HW Prop. Name: Collision Detect:

Neuron ID: Node Specs: none

Name of emulator, SBC, or custom node

Bandwidth Utilization: Received: 932 Logged: 0

Figure 6-7. emulator_1 Target HW Copy Window

- 5 Select fields and enter new values as needed. The fields and their defaults are:

App HW Name

enter a 1- to 16-character unique hardware name. You can use letters, numbers, and underscores in the name. No spaces are allowed. (The name in this field matches the Hardware Name you enter in the node specification definition, as described under *Defining Node Specifications*, later.)

A custom node is based on user-developed target hardware.

HW Type

choose the target hardware type: Emulator, SBC, or Custom. (The default is **Emulator**.)

■ Page 6-18 Addition

Location

Add the following to the field description of HW Type (top):

If you are modifying a definition, this field is read-only. If you wish to reuse the definition for some different hardware type, use the Copy button in the Navigator window to copy the definition to a new definition, then change the hardware type before saving the new definition.

for HW Type Emulator or SBC, enter the development station ID/slot number for the target hardware. Valid development station IDs are 1, 2, 3, and 4. Valid slot numbers are 2, 3, 4, 5, 6, and 7. (Slot 1 is used by the control processor.) The default ID/slot number for emulator_1 is 1/2. The default ID/slot number for emulator_2 is 1/3. If you have already configured other target hardware for the default ID/slot number, enter an available ID/slot number.

or

after an SBC has been installed in the development station, this field may be modified to 0/0. This indicates that the SBC may be removed from the development station.

or

for HW Type Custom, do nothing. The Location field becomes read-only and displays 0/0.

Channel

enter the channel name for the node. (The default is **default_channel**. If you do not use the default, you must create the channel definition before you enter its name in this field. See *Defining Channels*, in Chapter 10.)

Priority

enter the priority of the node from 0 to 127. 0 means no priority; otherwise, the lower the number, the higher the priority. (The default is 0. The value in this field must be equal to or less than the value for Num Priorities in the channel definition described under *Defining Channels*, in Chapter 10.)

HW Prop Name

enter the hardware properties name. (The default is **default_hw_props**. If you do not use the default, you must create the properties definition before you enter its name in this field. See *Defining Properties*, earlier.)

Collision Detect

choose YES to enable collision detection, or NO to disable. (The default is **YES**). (This field only applies if the selected channel has Collision Detection enabled.)

■ Page 6-19 Addition

Add the following fields for Target Hardware:

Override Xcvr G.P. Data? If this hardware uses a special purpose mode transceiver, and the channel definition allows nodes to override the general purpose data (see *Defining Channels*, in this supplement), choose Yes to override the channel definition with the data defined in the following field. Choose No to use the data defined by the channel. This field is read-only if the channel selected for this hardware does not use special purpose mode or if it does not allow nodes to override the general purpose data.

Xcvr General Purpose Data if the Override Xcvr G.P. Data field is Yes, enter the seven hex bytes of data that will be downloaded to the special purpose mode transceiver. These values will override the values defined for the channel. This field is read-only if the Override Xcvr G.P. Data field is No, and will display the values defined for the channel.

NEURON ID

is a read-only field. This field value is assigned when you first install the node, and may not be changed by the user.

Node Specs

is a read-only field. This field value is assigned when you define the node specifications for this node. (See *Defining Node Specifications*, later, for details.)

6 Select buttons as follows:

SAVE

creates a new target hardware object definition with the values in the fields. After saving, enter new values to create another target hardware object definition, or select CANCEL to return to the Navigator window.

NEXT

displays the values of the next target hardware object selected.

CANCEL

returns you to the Navigator window without creating a new definition.

Defining Application Images

The NEURON C Developer's Kit is required to use a NEURON C source file.

The LONTALK API is required to use a Host C definition file.

Every application node must have an application image definition in the object database. The node's application image can be defined by a NEURON C source file (.NC extension), object code files (.NO extension and .BIF extension), an external interface file (.XIF extension), or a host C definition file (.HC extension). The node's application image definition has the same name as its NEURON C program file minus the .NC extension, object code file minus the .NO extension, the external interface file minus the .XIF extension, or the host C definition file minus the .HC extension. An external interface file may be created by a node export (see *Exporting and Importing* in Chapter 7) or a SNVT import (see *Importing SNVT Information from the Node*, later in this chapter).

These steps may be skipped if the application image origin is source code. An application image definition is created automatically when a node specification is created using an undefined application image.

To define an application image, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the App Image button to display the existing application image definitions in the object list.
- 4 Select the Create command button to open the Application Image Create window (see figure 6-8).

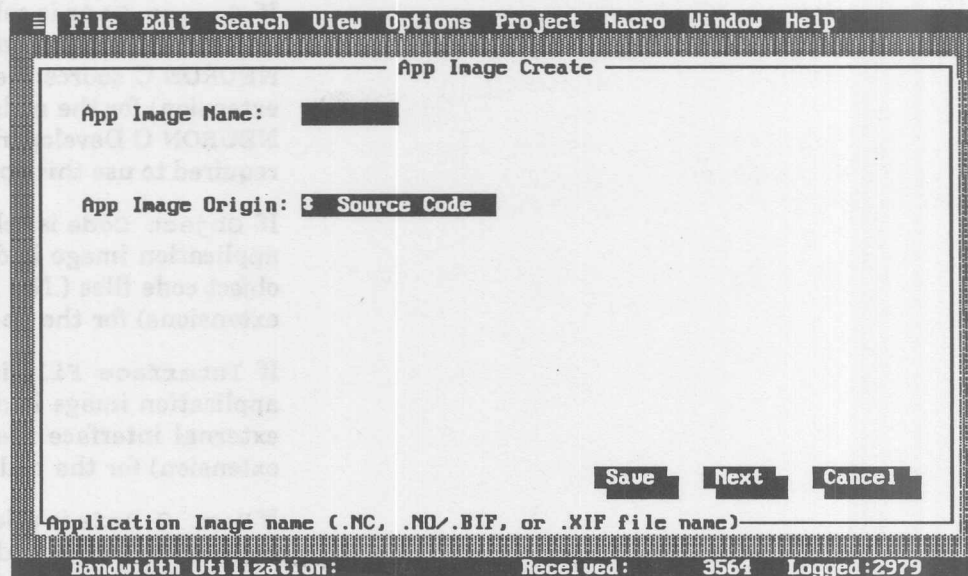


Figure 6-8. Application Image Create Window

- 5 Select fields and enter values as needed. The fields and their defaults are:

App Image Name

enter the 1- to 8-character name of the application image definition for the node. You can use letters, numbers, and underscores in the name. No spaces are allowed. This name must be the same name as the node's NEURON C program file, object code file, external interface file, or host C definition file, without the extension.

App Image Origin choose from the following options:

If Source Code is selected, the application image is defined by a NEURON C source file (.NC extension) for the node. The NEURON C Developer's Kit is required to use this option.

If Object Code is selected, the application image is defined by object code files (.NO and .BIF extensions) for the node.

If Interface File is selected, the application image is defined by an external interface file (.XIF extension) for the node.

If Host C Defs is selected, the application image is defined by a host C definition file (.HC extension) and an external interface file (.XIF extension). The name of the host C definition file is the application image name plus ".hc". An additional field allows entry of the input external interface file name if this option is selected. The LONTALK API is required to activate this option.

6 Select buttons as follows:

SAVE

creates a new application image definition with the values in the fields. After saving, enter new values to create another application image definition, or select CANCEL to return to the Navigator window.

CANCEL

returns you to the Navigator window without creating a new definition.

Defining Node Specifications

Every application node must have a node specification in the object database. The node specification specifies the application node's name, its associated application image definition, and its associated target hardware definition.

To define the node specifications for an application node, follow these steps:

- 1** Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2** Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3** Select the Node Specs button to display the existing node specifications in the object list.
- 4** Select the Create command button to open the Node Create window (see figure 6-9).

-or-

Select a node specification from the object list. Select the Copy command to open the Node Copy window.

Node Create

Node Name: App Image Name: ↓

Topology

Subnet 1: default_subnet ↓ Subnet 2: ↓

Sub/Node #: 0/ 0 Sub/Node #: 0/ 0

Auth Key 1: ff-ff-ff-ff-ff-ff Auth Key 2: ff-ff-ff-ff-ff-ff

Messaging	Hardware
Net Mgt Authenticate: <input type="text"/> No	Target HW: <input type="text"/> ↓
Max Free Buffer Wait: <input type="text"/> 10 s	Location: <input type="text"/>

Save Next Cancel

Node name must be unique

Bandwidth Utilization: ■ Received: 26653 Logged: 5000

Figure 6-9. Node Specification Create Window

- 5 Most of the fields in this window are used when the node is installed in a development network, as described in Chapter 10, *Defining Development Networks*. Three of the fields are used for defining the application node independent of its network configuration.

The fields in the Node Specification Create window are:

Node Name	enter a 1- to 16-character unique name for the application node. You can use letters, numbers, and underscores in the name. No spaces are allowed. (The LONBUILDER software displays this name in the Application Node Name field of this node's target hardware definition as described under <i>Defining Target Hardware</i> .)
App Image Name	enter the application image name for the node. (The name in this field is described under <i>Defining Application Images</i> , earlier. If there is no application image with this name, one will be created automatically and the origin will be NEURON C source code.)
Target HW	enter the target hardware name of the emulator, SBC, or custom node. (The name in this field matches the App HW Name you entered in the node's target hardware definition, as described under <i>Defining Target Hardware</i> , earlier.)

6 Select buttons as follows:

SAVE	creates a new node specification with the values in the fields. After saving, enter new values to create another node specification, or select CANCEL to return to the Navigator window.
NEXT	displays the values of the next node specification object selected.
CANCEL	returns you to the Navigator window without creating a new specification.

Installing Application Node Target Hardware

Whenever you define a new target hardware object, it must be installed to ensure that the object database is consistent with the node hardware. You must also re-install target hardware whenever you cycle power on a node or change the property, channel, or target hardware definition of a node.

To install or re-install emulator or SBC target hardware in a development station, follow these steps:

- 1 Physically install emulators and SBCs as described in the *LONBUILDER Startup and Hardware Guide*.
- 2 Define the emulators' and SBCs' hardware property, channel, and target hardware parameters, as described in this chapter, if you have not already done so.
- 3 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 4 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 5 Select the Target HW button to display the existing target hardware definitions in the object list.
- 6 Select one or more target hardware objects to install.
- 7 Select the Install button to receive a message dialog box saying:

Command in Progress

When installation is complete, a message dialog box appears saying:

Command Complete

To install or re-install SBC target hardware on a remote network, follow these steps:

- 1 Install the SBC in a development station as described above.
- 2 Set the location field of the SBC target hardware to 0/0.
- 3 Physically install the SBC remotely from the development station as described in the *LONBUILDER Start-up and Hardware Guide*.

To install or re-install custom node target hardware, follow these steps:

- 1 Physically install the custom node as described in the *LONBUILDER Startup and Hardware Guide*.
- 2 Define the custom nodes' hardware property, channel, and target hardware parameters, as described in this chapter, if you have not already done so.
- 3 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 4 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 5 Select the Target HW button to display the existing target hardware definitions in the object list.
- 6 Select one or more target hardware objects to install.
- 7 Select the Install button to display the dialog box message:
To register press service pin. Proceed?
- 8 Select YES to display the dialog box message:
Waiting for the custom node's service pin event

■ Page 6-27: Addition

Add the following set of SBC installation procedures after step 3: -
To change the communications parameters on an SBC once it has been installed on a remote network, follow these steps:

- 1 Select the SBC target hardware to install.
- 2 Select the Install button to display the dialog box message:
This hardware has been previously installed.
Use the NEURON ID currently in the database?
- 3 Select Yes to display the dialog box message:
Do you want to install communications parameters?

Follow the instructions for installing new communications parameters on a custom node, starting with step 10 on page 6-28 of the *LONBUILDER User's Guide*.

■ Page 6-27 Correction

Step 7 should read as follows:

- 7 When installing a custom node, if you have already installed it at least once, you will first see a dialog box with the message:

This hardware has been previously installed.
Use the NEURON ID currently in the database?

If the NEURON ID has not changed since you last installed the node (you have not swapped the node hardware), you may select Yes to skip the service pin registration and proceed to the optional updating of communications parameters (step 10). If the node hardware has been swapped, or you are not sure which physical node was used when this target hardware was last installed, select No and proceed with the normal service pin registration. In this case, the following message box will be displayed:

To register press service pin. Proceed?

Answer Yes to this question before activating the node's service pin.

- 9 Press the service pin on the custom node to be installed. If the network manager does not receive the service pin message, or if you take too long to press the service pin (about 10 seconds), the following message is displayed:

Service pin message not received.

If you get this message, check that a communications path exists from the custom node to the network manager, and begin again from step 7.

If you press the service pin within the allotted time, the following message is displayed:

Do you want to install communications parameters?

- 10 Select buttons as follows:

When changing communication parameters, the new parameters do not take effect until the node is reset, or the power is cycled on the node.

YES

writes the communication parameters to the target hardware's EEPROM memory and displays the message: Command Complete. Installation proceeds to the next selected target hardware object. This option should only be used when changing the custom node's transceiver or priority. After selecting this option, power down the custom node and then change the transceiver.

This option will also change the input clock parameter for the node, as specified in the Hardware Properties screen. If this value is incorrect, the node will be unable to communicate. NEURON 3150 CHIP-based nodes may be recovered using the EEBLANK image. See *Building Custom Nodes* in chapter 7.

CAUTION: Ensure that you have selected the correct input clock rate before changing the communication parameters for a node.

NO

does not write the communication parameters to the target hardware's EEPROM memory, but does display the message: Command Complete. Installation proceeds to the next selected target hardware. Select this option unless you are changing the custom node's transceiver or priority. (This option is the default.)

CANCEL

does not write the communication parameters to the target hardware object's EEPROM memory and cancels installation for the remainder of the selected target hardware objects.

The project manager can be used to simplify target hardware installation. To install nodes using the project manager, follow these steps:

- 1 Install the node hardware as described in the *LONBUILDER Startup and Hardware Guide*.
- 2 Open the Project pull-down menu.
- 3 Select one of the following commands:

Automatic Install installs the target hardware of all nodes defined in the object database that require it.

Install All

installs the target hardware of all nodes defined in the object database, regardless of current status.

Modifying Application Node Objects

To modify an application node object, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the **App Node** button to list the application node object type selectors—**Properties**, **Target HW**, **App Image**, and **Node Specs**.
- 3 Select either **Properties**, **Target HW**, **App Image**, or **Node Specs**. Existing objects for the selected type display in the object list.
- 4 Select one or more of the displayed objects.
- 5 Select the **Modify** command button to open the **Modify** window with the values for the first object displayed in the fields. (The **Modify** window for **Properties** looks like the window in figure 6-3, for **Target HW** looks like the window in figure 6-4, for **App Image** looks like the window in figure 6-5, for **Node Specs** looks like the window in figure 6-6.)
- 6 Select the fields you want to modify, and enter new values.
- 7 Select buttons as follows:

SAVE	modifies the object with the values in the fields. After saving, select NEXT or CANCEL .
NEXT	displays the values of the next object selected.
CANCEL	returns you to the Navigator window without modifying the object.

Deleting Application Node Objects

When you delete a node's Target HW or Node Specs definition, the node can no longer be loaded. If you attempt to load a partially deleted node, you will receive an error message. However, if the node was previously loaded, and its application is running, deleting the node will not stop the application.

To delete hardware properties or a channel definition for a node, you must first delete the node's hardware definition, or modify it so that it no longer contains the channel name or hardware properties that you wish to delete.

To delete an application node object, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selector—Properties, Target HW, App Image, and Node Specs.
- 3 Select either Properties, Target HW, App Image, or Node Specs. Existing objects for the selected type display in the object list.
- 4 Select one or more of the displayed objects.
- 5 Select the Delete command button.

Standard Network Variable Types (SNVTs)

The LONTALK protocol includes a predefined set of Standard Network Variable Types (SNVTs) to promote interoperability between LONWORKS products. SNVTs promote interoperability between nodes by providing a well-defined interface for internode communication. A node may be installed in a network and logically connected to other nodes in the network as long as the data types match.

Interoperability simplifies installation of nodes into different types of networks by keeping the network configuration independent of the node's application. For further explanation of SNVTs, please refer to the *NEURON C Programmer's Guide* and the *SNVT Guide*.

The Node Query Window

The Node Query window allows you to look at and import the definitions of network variables for a node. Network variable information can only be read when the node selected is installed and has an application loaded. Otherwise, an error message will indicate that the network variable information cannot be read.

The Node Query window contains a scrollable list of the network variables in the node, along with their types and some of their characteristics. If a network variable is not a known SNVT, it will be imported as Typeless. This will cause binding operations to ignore type checking for the variable. Any variables which are SNVTs will still work as expected.

The Self-Identification (SI) and Self-Documentation (SD) features determine how much information is available through Node Query. Refer to the *NEURON C Programmer's Guide* for an explanation of the compiler directives that control the SI and SD features.

The network manager cannot assure that Typeless variables are the correct size for a particular connection setup.

When the Node Query window is first opened, the first network variable in the list will be selected by default. The network variable information, for the selected network variable, is displayed in a subwindow below the list of names (see figure 6-10).

Querying Network Variable Information

To query network variable information, follow these steps:

- 1** Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2** Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3** Select the Node Specs button to display the existing application node specifications in the object list.

or

Select the Target HW button to display the existing target hardware objects in the object list.
- 4** Select one or more of the displayed nodes. A check mark (✓) appears to the left of the selected nodes or Target HW objects.
- 5** Select the Query command button to open the Node Query window. If the node(s) selected are not installed and loaded, an error message appears indicating that the network variable information cannot be read. This information is read-only (with the exception of the Network Variable Name field.) A maximum of five network variables can be displayed in the list at a time. Use the scroll bar to the right to view all available network variables. If any information is not available, its field is left blank.
- 6** To view another network variable, move the cursor to the left of that network variable and click the mouse. A check mark (✓) appears to the left of the selected network variable.

File Edit Search View Options Project Macro Window Help

Node Query - temp_mon_remote on sbc_remote_1

NU Name	Type	Measurement	Units	Size
nu_temp	SNUT_temp	Temperature	degrees Celsius	2
nu_mute	SNUT_leu_disc	Discrete level	*	1
nu_set_point	SNUT_temp	Temperature	degrees Celsius	2
nu_display_units	Typeless			

Synchronous	No	Change when offline	No
Polled	No	Config Class	No
Direction	Output		
Service Type	Config	Ackd	
Priority	Config	No	
Authentication	Config	No	
Max Message Rate (messages/sec)	Unspec	NU Name	nu_temp
Avg Message Rate (messages/sec)	Unspec	Array Size	0

Netvar SD String

Current temperature

Import Next Cancel

Bandwidth Utilization: Received: 1669 Logged: 1669

Figure 6-10. The Node Query Window

7 Select buttons as follows:

Import

imports the network variable definitions and saves the name changes. Import creates an external interface file (.XIF extension) in the current working directory. The external interface file has all the information that you were able to read from the node. Use this option when importing nodes from another source into your development network.

Next	displays the next node's list of network variables.
Cancel	returns you to the Navigator. If you Cancel after editing network variable names without importing the node, the name changes will be lost. You will be prompted prior to this loss and given the opportunity to recover.

Editing Network Variable Names

If the imported node's application program was built with the `enable_sd_nv_names` compiler directive, the network variable names in the Node Query window will be the same names used in the original program. If `enable_sd_nv_names` was not used, the network variable names will be `nv_0`, `nv_1`, etc. These names may be edited for readability.

To edit a network variable name, follow these steps:

- 1 Follow steps 1 to 5 under *Querying Network Variable Information*, earlier.
- 2 Move the cursor to the Network Variable Name field and type in the new name. Press **Enter** and the new name will appear in the network variable list at the top of the screen.
- 3 Repeat step 2 to rename other network variables on the same node.
- 4 For the name changes to take effect, import the external interface file as described under *Importing SNVT Information From the Node*, later in this chapter.
- 5 To view another node, select **NEXT** to display the next selected node's list of network variables. To return to the Navigator, refer to step 7 under *Querying Network Variable Information*, earlier.

The name change is not reflected in the node.

Describing Network Variable Parameters

To view the Network Variable Parameters:

- 1 Follow steps 1 to 5 in the *Querying Network Variable Information* section earlier. If the network variable is an array, the detail fields describe only the first element of the array.
- 2 Select a network variable from the network variable list at the top of the Node Query window. Details for the selected network variable are displayed at the bottom of the window.

These fields are listed below. (Refer to the *NEURON C Programmer's Guide* for more detailed descriptions.)

Synchronous	specifies that all values assigned to this network variable must be propagated, and in their original order.
Polled	specifies that the value of the output network variable is to be sent only in response to a poll request.
Change only if offline	is used to signal to a network management tool that a node should be taken offline before an update can be made to the network variable. This is commonly used with a config class network variable.

CONFIG class	specifies a const network variable in EEPROM that can be changed only by a network management message. This class of network variable is used for application configuration by a network management tool or a network controller.
Direction	declaring a network variable as either input or output (relative to the target node).
Service Type	specifies the LONTALK protocol service used for updating this network variable. The possibilities are ackd, unackd, and unackd-rpt.
Priority	specifies whether an update to this network variable has priority access to the communication channel. When a priority network variable is updated, its value will be propagated on the network within a bounded amount of time if the node is configured to have a priority slot.
Authentication	is a special form of an acknowledged service between one writer node and from 1 to 63 reader nodes. Authentication is used to verify the identity of the writer node.
Config/no config	for service type, priority, and authentication. This specifies that the parameter may be altered over the network by a network management tool.

Max Message Rate	the maximum message rate in messages per second that the associated network variable is expected to transmit. The maximum value allowed is 18780.
Avg Message Rate	the average message rate in messages per second that the associated network variable is expected to transmit. The maximum value allowed is 18780.
Network Variable Name	the name of the current network variable that you are viewing. If you did not specify the compiler directive <code>enable_sd_nv_names</code> , the system will generate names in the form of <code>nv_0</code> , <code>nv_1</code> , <code>nv_2</code> , etc. In either case, this field may be edited to change the name.
Array Size	If this network variable is identified as an array by the SNVT information, the array size will be displayed, and the detail fields describe the first element of the array. If not identified as an array, 0 will be displayed in this field.
SD String for netvar	the description of this network variable if the SD String was specified in the node's application program.
3 To view another node, select NEXT to display the next selected node's list of network variables. To return to the Navigator, refer to step 7 under <i>Querying Network Variable Information</i> , earlier.	

Importing SNVT Information from the Node

To import SNVT information from the node, follow these steps:

- 1 Follow steps 1 to 5 in the section *Querying Network Variable Information*, earlier.
- 2 Select the **Import** button. Import creates an external interface file (.XIF extension). The external interface file has all the information that you are able to read from the node.

To prepare to build using an external interface file (.XIF extension):

- 3 Create an application image definition for the node. Specify the imported external interface file as the App Image name and Interface File as the App Image origin. The App Image name for the associated node is used for the file base name.
- 4 Assign the application image file to a Node Spec.
- 5 Assign the Node Spec to a target hardware object. When you build the project now, the information from the external interface file is made available to the database, and connections can be created to the imported node. (Refer to Chapter 7 for detailed build information.)

7

Building Application Nodes

This chapter describes how to compile, link, and load NEURON C programs. Also included are instructions for building custom nodes, exporting and importing nodes and application images from one LONBUILDER system to another, and for exporting a node from a LONBUILDER system to a network management tool.

Selecting a Project Build Command

The LONBUILDER Project Manager is used to create and load application images, and to configure network images based on the definitions contained within the object database. The Project pull-down menu contains the build commands used to invoke the project manager and build an application. Use the build commands to:

A build command also connects nodes' network variable and message tags, if these connections have been defined. See Building the Network Image, in Chapter 11, for details.

- Compile NEURON C code and link all library and system image files needed to run a project's application nodes. The compilation process generates object files, which are placed in the project directory. You can build your application's nodes as they are implemented, you can build all your application's nodes with a single build command, and you can rebuild nodes at any time.
- Optionally, load all system and application images in the project directory to the application nodes' target hardware. The project parameters (described under *Configuring LONBUILDER Project Parameters*, in Chapter 4) determine if the build command loads executables after compiling, or stops after compiling.

Before you issue a build command, you must:

- Define project parameters as described in Chapter 4.
- Create NEURON C programs as described in Chapter 5 and in the *NEURON C Programmer's Guide*, or import the nodes' external interface files as described in Chapter 6.
- Define application nodes as described in Chapter 6.

To build the application nodes defined in a project's object database, follow these steps:

- 1 Open the Project pull-down menu.

2 Select one of the following commands:

The Automatic Build, Automatic Load, and Build All commands also invoke the binder to connect nodes' network variables and message tags, if these connections have been defined. See Building the Network Image, in Chapter 11, for details.

Automatic
Build

Selecting this command recompiles and relinks the project's NEURON C and include files that require it. The Automatic Build command calculates all dependencies and recompiles files that have changed since the previous build. If the Load After Build project configuration parameter is on, the Automatic Build command loads the system and application images and configures the network images after compiling and linking, as needed. If the Start After Load project configuration parameter is on, nodes that are loaded by the Automatic Build command are put online, otherwise they are left in the reset/halt state (debuggable nodes) or offline (non-debuggable nodes).

Build All

Selecting this command recompiles and relinks all NEURON C and include files defined in the project's object database. If the Load After Build project configuration parameter is on, the Build All command loads all of the system and application images and configures all network images after compiling and linking.

Automatic
Load

Selecting this command performs the same actions as the Automatic Build command, except that the application nodes are automatically loaded and configured regardless of the setting of the Load After Build project configuration parameter.

When you select a build command, the Project Manager window opens (see figure 7-1).

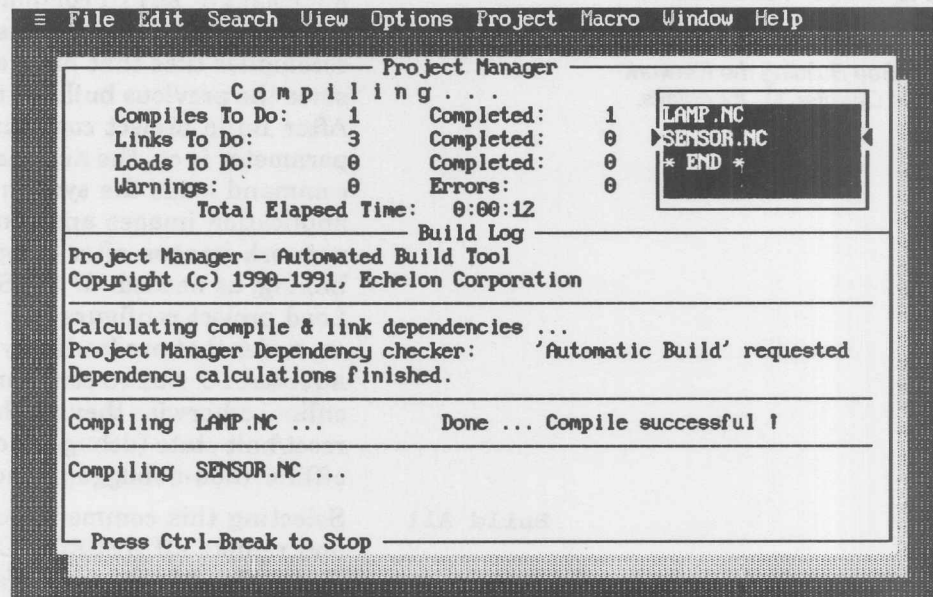


Figure 7-1. Project Manager Window

As the build procedure progresses, the Project Manager window displays:

- Compilation information, including the .NC files being compiled, how many compiles, links, and loads are left to do, and how many compiles, links, and loads are done.
- The build log, listing the warnings and errors as they occur, and stating if the build is successful.

Compilation stops after it encounters a warning or error, if you have set the Stop Build On Warnings or Errors option to **On**. Compilation stops after it encounters a warning or error and invokes the editor, if you have set the Invoke Editor on Build Stop option to **On** and if the warning or error is applicable to the NEURON C program. Other warnings or errors do not invoke the editor. Both of these options are described under *Configuring LONBUILDER Project Parameters* in Chapter 4.

When the build procedure halts to invoke the editor, you receive the prompt:

```
Ready to Invoke Editor (Ctrl-Break will  
cancel) . . .
```

You can press the following keys:

- Any key to open the editor with the build log and the NEURON C file that caused the error displayed in the Editor windows.
- *Ctrl-Break* to skip invoking the Editor. When you skip the editor, you receive the message:

```
Invocation of the Editor has been cancelled.
```

See *Using the Build Log with the Editor*, in Chapter 5, for more information.

Whether stopped for warnings or errors or successfully completed, the build process places a copy of the build log (named BUILD.LOG) in your current working directory. The most current build log overwrites any existing build log.

You can view the build log file at any time by opening the editor window, as discussed in Chapter 5, and then opening the BUILD.LOG file.

If a node's configuration is changed either by the node itself or by an external network management tool, the LONBUILDER system will not know of these changes. Subsequent Automatic Builds (or Automatic Loads) may inadvertently corrupt the node's configuration if the node is not installed in a development station. A manual load or a Build All should restore the node.

If the node's state is changed to *unconfigured* or *applicationless*, either by an external network management tool, by the node's application program, or due to a configuration or application checksum error, an Automatic Build (or Automatic Load) will not load the node unless the LONBUILDER system knows about some configuration change that needs to be made. In any case, if the node is not installed in a development station, the node's configuration may be corrupted by an automatic load. A manual load (from the Navigator screen) or a Build All should restore the node.

Compiling NEURON C Programs

In addition to the build commands, the LONBUILDER Project Manager allows you to compile individual NEURON C programs to see if they compile without errors. However, you cannot load the resulting object code, because it has not been linked. You can only load object code created with an Automatic Build, Automatic Load, or Build All command from the Project pull-down main menu. These build commands are described earlier in this chapter under *Selecting a Project Build Command*.

Before you issue a compile command, you must:

- Define project parameters as described in Chapter 4.
- Create NEURON C programs as described in Chapter 5 and in the *NEURON C Programmer's Guide*.

To compile from the Editor window, follow these steps:

- 1 Move the cursor to the editor text window containing the NEURON C source code you want to compile. (How to open the editor and edit text is described in Chapter 5, *Editing Files*.)
- 2 Open the Project pull-down menu.
- 3 Select the Compile File command from the Project pull-down menu.

If warnings or errors are encountered, the compilation stops as specified by the Stop Build On options, described under Configuring LONBUILDER Project Parameters, in Chapter 4.

To compile from the Navigator window, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the File button to list the file object type selectors—NEURON C and Include File.
- 3 Select the NEURON C button to display the existing NEURON C filenames in the object list.
- 4 Select one file in the object list. A check mark (✓) appears in front of the selected file.
- 5 Select the Compile command button. The Project Manager window (see figure 7-1, earlier) opens and displays the compilation information for the file.

Loading Built Nodes and Routers

The Network Manager node must be installed on the appropriate channel before you can load remote nodes.

Emulators, SBCs, and routers that are installed on the backplane (and whose Target Hardware location is non-zero) are loaded over the backplane. All other nodes and routers are loaded using the network manager, and require a valid communication path from the network manager to the target nodes and routers.

You can set project configuration parameters, as described in Chapter 4, to load nodes and routers with a build command. Or, you can stop a build before loading. When you don't include loading in a build, there are two methods you may use to load nodes:

- Automatic Loading
- Manual Loading

Before you load images onto one or more nodes, you must:

- Define project parameters as described in Chapter 4.
- Create NEURON C programs as described in Chapter 5 and in the *NEURON C Programmer's Guide*.
- Define and install target hardware.
- Define application nodes as described in Chapter 6.
- Define routers as described in Chapter 9.
- Issue a build command as described earlier under *Selecting a Project Build Command*.

Loading Application Nodes and Routers Automatically

To load application nodes and routers automatically, follow these steps:

- 1 Open the Project pull-down menu.
- 2 Select the Automatic Load command from the Project pull-down menu to download existing images to the project's configured nodes. If you select this command, any necessary compiles, links, or binds are done prior to loading.

After loading, application nodes will be put online if the Start After Load option is set. Otherwise, application nodes are left in the reset/halt state (debuggable nodes) or offline (non-debuggable nodes). Routers are always put online after loading.

Loading Application Nodes Manually

To load application nodes manually, follow these steps:

- 1 Press F2 to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the Node Specs button to display the node specifications or the Target Hardware button to display the target hardware definitions for application nodes.
- 4 Select the nodes you want to load. A check mark (✓) appears in front of the selected nodes.
- 5 Select commands as follows to load the selected nodes:

For MIP-based nodes, when you select the Load option, a Load/Start will automatically be performed.

Load/Start

loads the node and starts execution. The application is put online.

Load

loads the application. If the node is debuggable, the node is left in the reset/halt state. Otherwise, the node is left offline.

Loading Routers Manually

To load routers manually, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Router button to list the application object type selectors—Target HW and Node Specs.
- 3 Select the Node Specs button to display the node specifications or the Target HW button to display the target hardware definitions for routers.
- 4 Select the routers you want to load. A check mark (✓) appears in front of the selected routers.
- 5 Select the Load/Start command button to load and start the router. Routers are always put online after loading.

Loading Nodes Through Routers

Remote nodes are loaded over the network, using the network manager to update the application and network images. The network manager may require the services of one or more routers to communicate with a given node. The target node, and all of the routers in the communication path between the network manager and target node, must have a domain in common; repeaters do not require a common domain. The network manager confirms that the configuration of the routers and target nodes share a domain. If they do not, an error message is displayed.

All of the routers that appear in the communication path must be loaded and online prior to loading the target nodes. If an automatic load is chosen, the project manager will load all necessary routers in the correct order. If a manual load is selected, the selected routers will be loaded in the correct order. However, the network manager will not verify that all required routers are running and online prior to the manual load. This may result in load failures.

Certain changes to network configuration can also cause load failures, primarily illegal domain changes and moving a subnet from one side of a router to another.

Illegal domain changes for a node occur when all of the following conditions are met:

- The node is not on the backplane.
- The node was configured with network management authentication turned on when it was loaded last. (Refer to Chapter 10 *Defining Node Specifications*.)
- The node is not on the same channel as the network manager.
- The node's domain has been changed.

There are three ways to load a node once an illegal domain change has occurred:

- Change the domain back to the domain used for the last successful load, and turn off network management authentication.
- If the node is an emulator or SBC, move the node onto the backplane. Then, re-install and re-load the node.
- Move the node to the network manager's channel.

Moving a subnet from one side of a router to another causes problems when learning or configured routers are used. To avoid this problem, temporarily change your configured or learning routers to bridges or repeaters and reload everything that way. It is illegal to have the same subnet on both sides of a learning or configured router.

Exporting and Importing

The LONBUILDER software produces a number of files internally that may be exported by the user. Some of them are used with a PROM programmer for building custom nodes. Others may be imported by another LONBUILDER system or by a network management tool. The table in Figure 7-2 shows all the different types of files that may be exported, along with their file extensions and their uses.

LONWORKS Name	File Extension	File Use
NEURON ROM Image	.NRI	ROM image for a 3150-based custom node. Used as an input file to a PROM programmer.
NEURON EEPROM Image	.NEI	External EEPROM image for a 3150-based custom node, or internal EEPROM image for a 3120-based custom node. Used as input to an external EEPROM programmer or to a NEURON 3120 CHIP programmer.
Downloadable Image File	.NXE	Downloadable application image. Used by a network management tool to download an application image over the network.
External Interface File	.XIF	External interface file. Defines all external interfaces to a node. Used by a network management tool for defining connections to a node.
NEURON Object	.NO	Compiled object code for a NEURON C program. Used to export an application program to another LONBUILDER system without transferring source code.
Connection Information File	.BIF	Connection information. Used with the corresponding NEURON object file for exporting an application program to another LONBUILDER system.

Figure 7-2. LONBUILDER Exported Files

The NEURON ROM Image (.NRI) File

The NEURON ROM Image (.NRI) file contains the NEURON CHIP firmware and some or all of the application code and data. This file is only used for building NEURON 3150 CHIP-based custom nodes, using a PROM programmer to program one or more PROMs that will be present on the node. The node's ROM code is contained in this single file. If you are building a node that uses multiple PROMs and the programmer you are using requires an individual file for each PROM to be programmed, use the LONBUILDER Editor to break the NEURON ROM Image file into separate files. When exported, the system ROM portion (the first 16K) also contains a copy of some or all of the on-chip EEPROM image. The extent of this copy is controlled by the Firmware State field when the file is exported. This field is described later in this chapter. This copy will be written to the on-chip EEPROM on initial power-up of an uninitialized NEURON 3150 CHIP, and determines the state that the node comes up in. See *Building Custom Nodes*, later in this chapter.

The NEURON EEPROM Image (.NEI) File

The NEURON EEPROM Image (.NEI) file is used for building both NEURON 3150 CHIP and NEURON 3120 CHIP-based custom nodes. Its contents depends on which chip model is being used. In both cases it contains EEPROM data. For a NEURON 3150 CHIP-based node, it is the application code and data that resides in off-chip EEPROM or non-volatile RAM, if any. In this case the NEURON EEPROM Image file is used with a PROM programmer to program the external memory chips. For a NEURON 3120 CHIP-based node, it contains some or all of the on-chip EEPROM image, in a special format for use only with a NEURON 3120 CHIP programmer. Like the NEURON ROM Image file used with a NEURON 3150 CHIP-based node, the extent of the on-chip EEPROM data, in a NEURON EEPROM Image file used with a NEURON 3120 CHIP-based node, is controlled by the Firmware State field when the file is exported. See *Building Custom Nodes*, later in this chapter.

The Downloadable Image (.NXE) File

The Downloadable Image (.NXE) file contains the portion of an application that can be downloaded to a node over the network. This is all the code and data that resides in on- or off-chip, writeable, non-volatile memory (EEPROM or NVRAM). It is in a format that may be used by a network management tool to load a node's application in a production network. For example, the LONMANAGER API will accept a .NXE file in Intel Hex format and load a node with it.

The External Interface (.XIF) File

The External Interface (.XIF) file contains all the connection information from a connection information (.BIF) file, plus selected hardware parameters pertaining to the node. It can be used by a network management tool when making connections to a pre-loaded node. For example, the LONMANAGER API can load the connection information from an external interface file and use it to make connections. An external interface file may also be imported by a LONBUILDER system when using pre-loaded nodes on the development network. See *Importing External Interface Files*, later in this chapter. An external interface file must also be used when building a LONTALK API-based node. In this case the external interface file must be exported from a MIP-based node. An application image of type Host C Defs is then created and the name of this external interface file is entered where indicated. See *Defining Application Images* in Chapter 6. An external interface file can also be created by using the Query command to import a node's Standard Network Variable information. See *Importing SNVT Information from the Node* in Chapter 6.

■ Page 7-14 Addition

Add the following to the external interface file (.XIF) description:

LONBUILDER 2.2 exports external interface files in version 3 format. If the version 2 format is needed, use the utility program XIF3TO2.EXE to convert the format of the file. Version 2 format may be required by some installation tools, for example, NETMAKER 1.0. See the changes to page D-4 of the *LONBUILDER User's Guide* for a description of XIF3TO2.

The NEURON Object (.NO) and Connection Information (.BIF) Files

The NEURON object (.NO) and the connection information (.BIF) files are always exported and used as a pair. They contain the object code and connection information generated from the compilation of a NEURON C program. These files are only used for importing into another LONBUILDER system, which would be done to provide a user with a linkable version of your application without giving them the source code. See *Importing Application Image Files*, later in this chapter.

The Components of an Exported Node

A node that is moved from a LONBUILDER development network to another development or production network is an exported node. The destination network imports the exported node, possibly along with one or more of the exported node files. An exported node may have one to three components:

- An SBC or custom node hardware
- An external interface (.XIF) file. This file is not needed if the node uses Standard Network Variable definitions exclusively. See *Importing SNVT Information from the Node* in Chapter 6.
- A downloadable image (.NXE) file. This file is not needed if the node is pre-loaded.

Figure 7-3 shows an overview of exporting and importing a node.

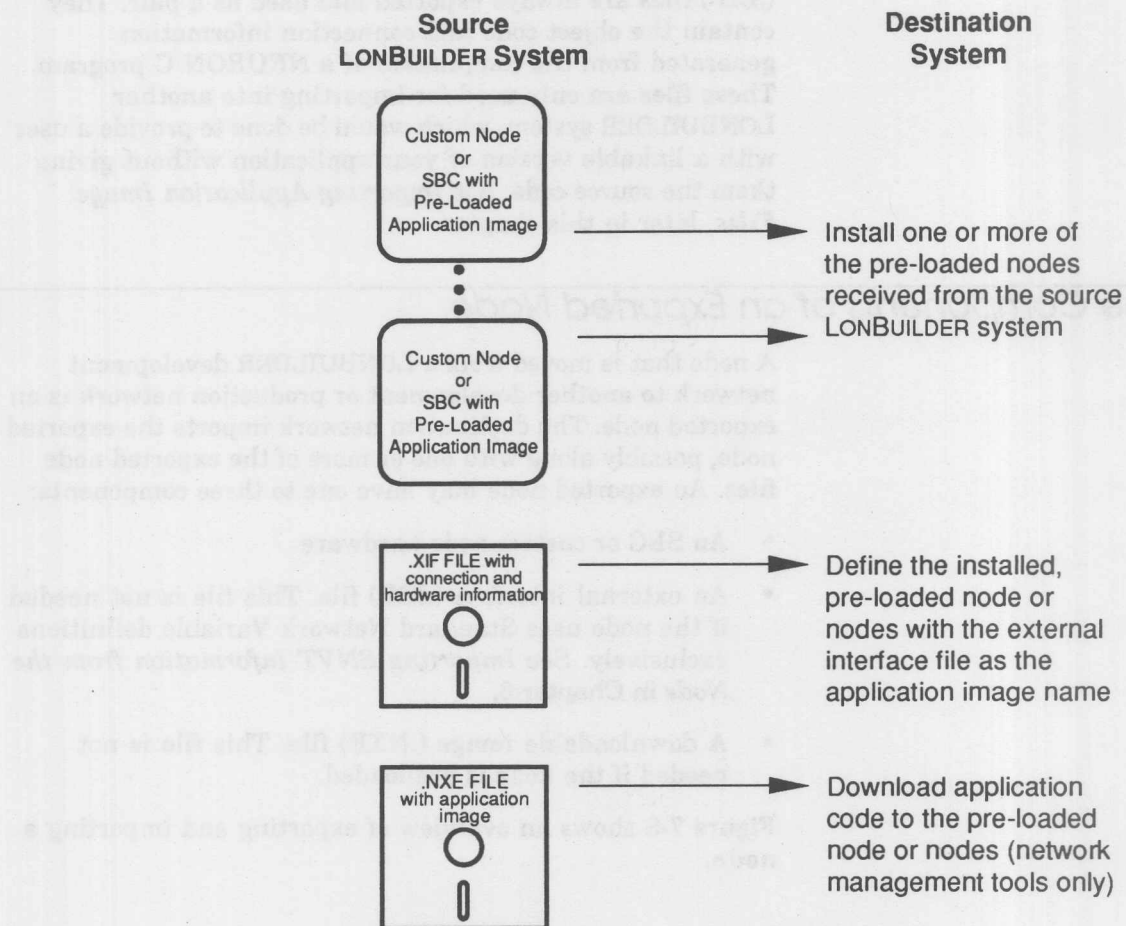


Figure 7-3. Exporting and Importing a Node

Exporting Node Files

A node file is one of the following four types: NEURON ROM Image (.NRI), NEURON EEPROM Image (.NEI), downloadable image (.NXE), or external interface (.XIF).

To export one of the node files, follow these steps:

- 1 Create the node's application image. To do this:
 - Write the NEURON C application program for the node or import application image files for the node, described under *Importing Application Image Files*, later.
 - Define the node as described in Chapter 6, *Defining Application Nodes*. The node's hardware type should normally be Custom.
 - Issue a build command as described earlier under *Selecting a Project Build Command*. Note that while a build is required, it is not necessary to load a node before exporting its files. In fact, the node hardware does not even need to exist. This will also be the case when first creating a custom node.
- 2 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 3 Select the App Node button to list the application object type selectors—Properties, Target HW, App Image, and Node Specs.
- 4 Select the Node Specs button to display the existing node specifications in the object list.

- 5 Select the node or nodes to be exported. A check mark (✓) appears in front of the selected nodes.
- 6 Select the More command button to access additional options. Select the Export command button to open the Node Export window (see figure 7-4). The name of the first selected node is displayed in the title bar.

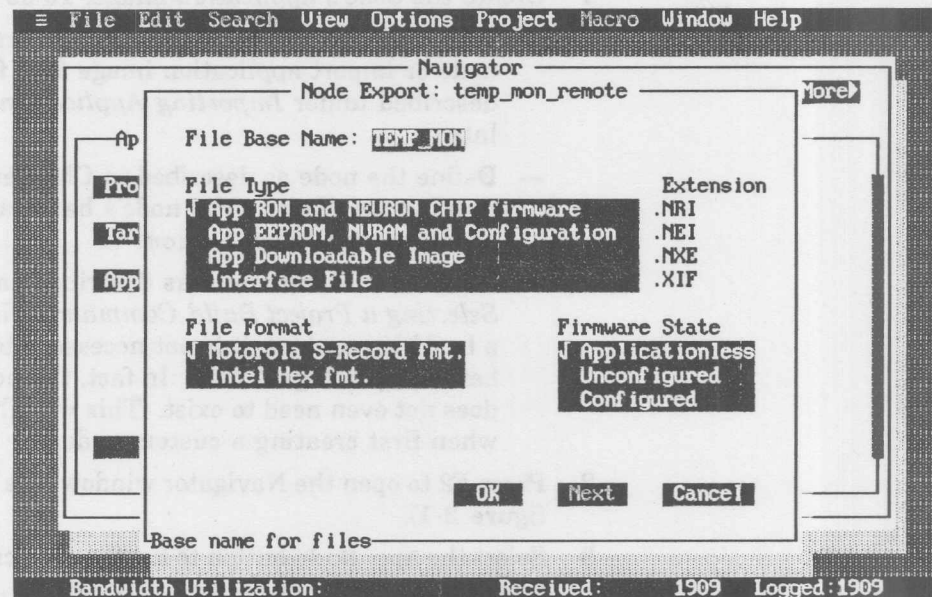


Figure 7-4. Node Export Window

- 7 Select Base File Name in the Node Export dialog box and enter a 1- to 8- character unique name for the node file. You can use letters, numbers, and underscores in the name. No spaces are allowed. (The default is the name of the selected node truncated to its first eight characters.)

8 Select one of the File Types. A check mark (✓) indicates the selected option. The four options are: App ROM and NEURON CHIP firmware; App EEPROM, NVRAM, and Configuration; Downloadable Image; and Interface File. The corresponding file extension will be appended to the file when the file is created. For all file types, except Interface File, the node you are using must be defined as a custom node. Refer to the introduction of *Exporting and Importing*, earlier, for a description of the file types.

The App ROM and NEURON CHIP firmware option cannot be selected if your node uses a NEURON 3120 CHIP.

9 Select one of the File Format options. A check mark (✓) indicates that the field is selected. The options are Motorola-S Record and Intel Hex. This will determine the data format of the exported file. These options are not available if the file type is Interface File.

10 Select one of the Firmware State options. These options determine the initial state of the node, as well as how much of the node's EEPROM is initialized by the NEURON CHIP firmware when the firmware determines that the EEPROM must be initialized. These options are not available if the file type is Downloadable Image or Interface file. These options are not used if the NEURON CHIP firmware is version 2 (the version shipped with LONBUILDER release 2.0). See *Building Custom Nodes*, later, for information on how to choose one of these options. A check mark (✓) indicates the selected option. The options are:

Application
less

only communication parameters are loaded into the internal EEPROM. The NEURON CHIP then assumes the *Applicationless* state. Use this option to build application nodes that will be loaded after manufacture.

Insert step 9a between steps 9 and 10:

9a Select whether the node's domain(s) should be cloned (Yes or No). Selecting Yes modifies the domain IDs to allow generation of multiple nodes from the same image, sharing the same subnet/node address. These nodes will be unable to receive messages that use subnet node addressing. (See *Cloned Domain IDs* below.) Selecting No leaves the domain IDs unchanged, enforcing unique subnet/node addresses. (The default is No.) This option is not available unless the file type is App ROM and NEURON CHIP Firmware or App EEPROM, NVRAM, and Configuration. Only files in which the firmware state is Configured are affected.

Unconfigured communication parameters and the EEPROM portion of the application image are loaded into the internal EEPROM. The NEURON CHIP then assumes the *Unconfigured* state. Use this option to build application nodes that will be pre-loaded, but installed in the field.

Configured the entire internal EEPROM portion of the application and network image is loaded. The network image includes any connections that have been defined as described in Chapter 10. The NEURON CHIP then assumes the *Configured* state. Use this option to build application nodes for systems that do not require field installation. These nodes will be pre-loaded and pre-configured.

11 Select buttons as follows:

OK	exports the selected node file by placing it in the current working directory, and returns you to the Navigator window.
NEXT	displays the next node selected.
CANCEL	returns you to the Navigator window. If you have not already pressed SAVE, a node file is not created for the selected node.

Importing External Interface Files

A pre-loaded node has its application image already loaded in some form of nonvolatile memory.

To import an external interface file and a pre-loaded node on a destination LONBUILDER system, follow these steps:

- 1 Copy the external interface file (.XIF) from the source LONBUILDER system to the working directory on the destination LONBUILDER system.
- 2 On the destination LONBUILDER system, physically install the node hardware as described in the *LONBUILDER Startup and Hardware Guide*.
- 3 Define and install the node as described in Chapter 6, *Defining Application Nodes*. This includes defining the node's Target Hardware, Hardware Properties and Node Specification.
- 4 Create a new application image. Select fields and enter values as follows:

App Image Name	enter the 1- to 8-character name of the external interface file. (Do not enter the .XIF suffix.)
----------------	--

App Image Origin	cycle through this field until Interface File displays in the field.
------------------	--

- 5 Modify the Node Spec created in step 3. Select the App Image Name and enter the name of the application image created in step 4.

When you build the node, as described under *Selecting a Project Build Command*, earlier, the build does not compile, assemble, link, or load the node's application image, because the application image is already on the node. The build does copy the node's connection information from the external interface file into the object database. The connection information may be used to create connections to the node. The external interface file must remain in the working directory for all subsequent builds.

Exporting Application Image Files

An exported application image consists of a NEURON Object file (.NO extension) and a connection information file (.BIF extension).

To export an application image, follow these steps:

- 1 Create the node's application image. To do this:
 - Write the NEURON C application program for the node or import application image files for the node.
 - Define the node as described in Chapter 6, *Defining Application Nodes*.
 - Issue a build command as described earlier under *Selecting a Project Command*.
- 2 Press F2 to open the Navigator window (see Chapter 3, figure 3-1).
- 3 Select the App Node button to list the application object type selectors—Properties, Target HW, App Image, and Node Specs.
- 4 Select one or more application images.
- 5 Select the Export command button to open the Application Image Export window (see figure 7-5). The name of the first App Image selected is displayed in the title bar.

■ Page 7-23 Deletion

Delete the screen shown in Figure 7-5.

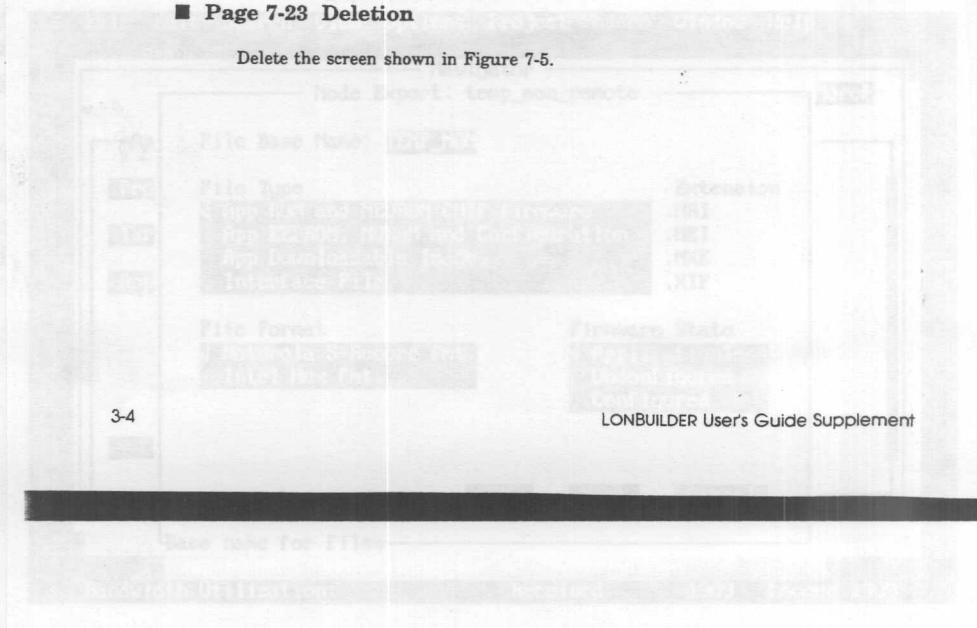


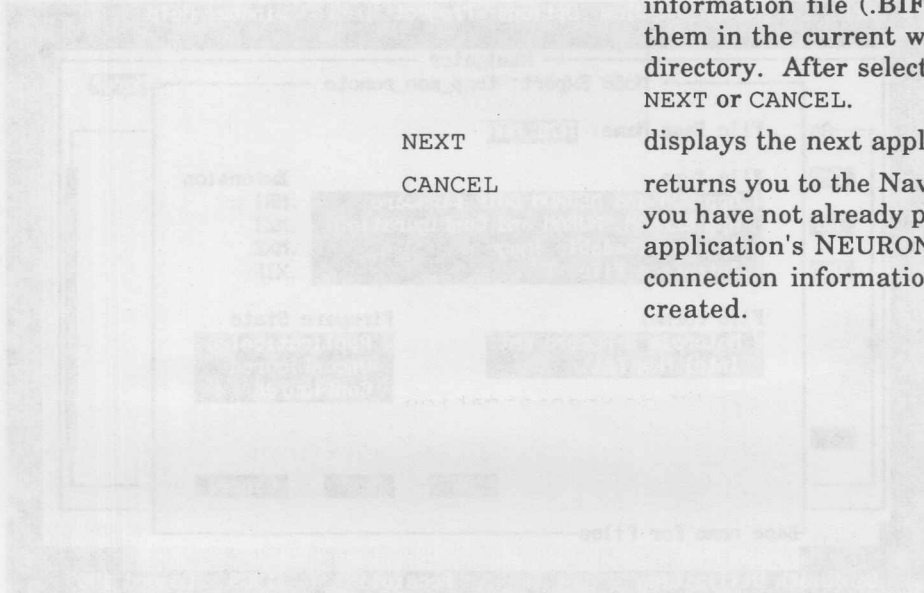
Figure 7-5. Application Image Export window

The connection information file contains the descriptions of a program's network variables and message tags. These descriptions are used by the binder to help create connections.

- 6 Select the File Name field and enter the 1- to 8-character base name for the application's NEURON object (.NO) and connection information (.BIF) files. You may use letters, numbers, and underscores in the name. (The default is the selected application name.) The .NO and .BIF extensions are automatically appended to the filenames when the files are placed in the current working directory.

7 Select command buttons as follows:

OK	exports the node's NEURON object (.NO) file and connection information file (.BIF) by placing them in the current working directory. After selecting OK, select NEXT or CANCEL.
NEXT	displays the next application selected.
CANCEL	returns you to the Navigator window. If you have not already pressed SAVE, the application's NEURON object file and connection information file are not created.



Importing Application Image Files

To import application image files to an installed node, follow these steps:

- 1 Copy the application image files (.NO and .BIF extensions) from the source LONBUILDER system to the working directory on the destination LONBUILDER system.
- 2 On the destination LONBUILDER system, physically install the node hardware as described in the *LONBUILDER Startup and Hardware Guide*.
- 3 Define and install the node as described in Chapter 6, *Defining Application Nodes*. This includes defining the node's Target Hardware, Hardware Properties, and Node Specification.
- 4 On the destination LONBUILDER system, create a new app image. Select fields and enter values as follows:

App Image Name	enter the 1- to 8-character base name of the exported application files. (Do not enter the .NO or .BIF extension.)
App Image Origin	cycle through this field until Object Code displays in the field.
- 5 Modify the Node Spec created in step 3. Select the App Image Name and enter the name of the application image created in step 4.

When you build the node, as described under *Selecting a Project Build Command*, earlier, the build does not compile or assemble, but does link and load the imported object code. The NEURON object and connection information files are also copied into the object database. The connection information may be used to create connections to the node. The NEURON object and connection information files must remain in the working directory for all subsequent builds.

Building Custom Nodes

A custom node is a NEURON CHIP-based module containing all the components required to function as a LONWORKS node. A custom node may be an actual end-product node, or a prototype design intended for analysis of the node's behavior in its working environment. In either case, the node resides outside of the LONBUILDER Development Station and is connected to the network through an appropriate transceiver.

A custom node based on a NEURON 3150 CHIP includes a NEURON 3150 CHIP, off-chip memory, a system clock, a transceiver, and I/O hardware. The off-chip memory contains the NEURON CHIP firmware, and possibly some or all of the application itself. A custom node based on a NEURON 3120 CHIP includes a NEURON 3120 CHIP, a system clock, a transceiver, and I/O hardware, but does not include any off-chip memory. The firmware and the application are contained entirely in the NEURON 3120 CHIP. Figure 7-6 shows a conceptual picture of two custom nodes. Refer to the *LONBUILDER Startup and Hardware Guide* for a description of the hardware components of custom nodes.

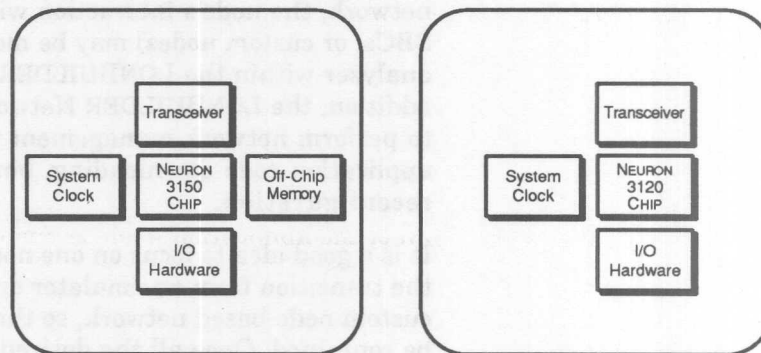


Figure 7-6. Two Custom Nodes

Guidelines for Creating Custom Nodes

To create a custom node, follow these steps:

- 1 Create an application program.
- 2 Build the node's hardware.
- 3 Program the node's memory.
- 4 Install the node on the network.
- 5 Load the node.

The application program should be tested and debugged as much as possible on an emulator, using the NEURON C Debugger. The programming or loading step may not always be required, depending on the NEURON CHIP model used, the external memory layout, and the type of network the node is installed in. See *Programming Custom Nodes* and *Loading Custom Nodes*, later in this chapter.

If a custom node is installed in a LONBUILDER development network, the node's interaction with other nodes (emulators, SBCs, or custom nodes) may be monitored using the protocol analyzer within the LONBUILDER Development Station. In addition, the LONBUILDER Network Manager may be used to perform network management functions such as application code downloading, binding, node testing, and reconfiguration.

It is a good idea to focus on one node at a time when making the transition from an emulator or SBC-based network to a custom node-based network, so that potential problems may be contained. Once all the desired nodes have been installed and are functioning properly, the network may be disconnected from the LONBUILDER Development Station and operated in a stand-alone mode. Prior to this point, however, it is a good idea to keep the LONBUILDER Development Station connected to the network so that its control and analysis tools may be used.

Target Network Types and Firmware State Selection

A custom node may be installed in a variety of networks. The following discussion considers three different types: a development network connected to a LONBUILDER Development Station; an open, dynamic, production network with a network management tool (but no LONBUILDER Development Station), and a closed, static, production network with no network management tool. The target network type will influence how the node is built. Several of the files exported from a LONBUILDER system which are used to build custom nodes require the selection of a firmware state: either applicationless, unconfigured, or configured (see *Exporting Node Files*, earlier in this chapter). This firmware state selection controls how the node will come up when it is installed in the network. Firmware state selection is largely determined by the degree of installation flexibility required.

Maximum flexibility is required when you must determine both a node's application program as well as its network configuration after the node is manufactured. For example, you may be manufacturing generic nodes that can be loaded with different application programs, depending on the options specified by a customer. This degree of flexibility is also useful during development, where the application program may change many times. In this case, build the custom node to start out as applicationless, and use the LONBUILDER Network Manager, or other network management tool, to load the application program and define the network configuration.

Less flexibility is required if a node's application program will be determined at the time of manufacture, but network configuration will still be done at the time of installation. This is the case when the node's entire application program is in ROM and therefore cannot be changed in the field. An example of field network installation would be a building lighting, heating, and security system, where nodes and connections are changed from time to time, various vendors' nodes are used, and the connections are not known at the time the nodes are built.

In this case, build the custom node to start in the unconfigured state (which implies that there is an application). When the node is powered up, the application will automatically start running, but there will be no connections to the node. The connections would then be made with a network management tool.

The least flexibility is required if a node's application program will be determined at the time of manufacture, and the network configuration will also be determined at the time of manufacture, or created using the NEURON C self-installation functions. A network management tool is therefore not required (but may be added in the future).

An example of this type would be a photocopier with an internal network, where all the nodes are predefined and created at the time of manufacture, and the connections are predetermined. In this case, build the custom node to start in the configured state. When the node is powered up, the application program will start running, and network connections will be established (or updated by the self-installation functions). One method of creating connections in this environment is to write applications that are self-installing. This allows simple connection choices to be made at installation time (through the use of hardware jumpers or selector switches), but it requires more sophisticated code and an understanding of the internal configuration structures.

If all the connections can be determined before the node is built, a simpler option is to define the connections before the ROM image is exported, and create the node in the configured state. When the node is programmed, all connection information is pre-loaded, so it will power up with the application running and all the connections in place.

If you create a node in the configured or unconfigured state, you can still choose to load a new application or new configuration at a later time. The selected firmware state only determines the node's initial state, subject to some restrictions described in the following section.

Programming Custom Nodes

■ Page 7-31 Addition

Add the following new section before *Programming Custom Nodes*:

Cloned Domain IDs

Part of a node's network configuration image is the domain ID for each domain of which the node is a member. A domain ID contains the subnet and node numbers assigned to the node for that domain. A node will usually reject any incoming messages sent by a node with the same subnet/node numbers, since normal connections require unique subnet/node assignments and a duplicate subnet/node ID represents a message that has been incorrectly forwarded to its originating channel. However, it is possible to modify domain IDs so that multiple nodes with the same subnet/node numbers can communicate. This is called cloning a domain. This technique is typically used in low-end self-installed nodes in a network where an unknown number of identical nodes can be used together, and no network management tool is available to assign unique IDs.

One method of cloning a domain is to use the NEURON C function `update_clone_domain()`. With this method, it is possible to clone one domain while maintaining a unique subnet/node address on the second domain. A second method is to indicate that domains should be cloned when the memory image file (.NRI for the NEURON 3150 CHIP or .NEI for NEURON 3120 CHIP) is exported. This method always clones any domain that is in use. Code space is saved because the library function is not required. This may be necessary on a NEURON 3120 CHIP with a large application.

The following consequences are associated with using cloned domains:

- Nodes using a cloned domain can no longer receive messages in that domain using subnet/node addressing. Some other addressing mode must be used (NEURON ID, group, or broadcast). Use only group and broadcast addressing for self-installed nodes since the use of NEURON ID addressing makes systems more difficult to maintain.
- The node cannot receive acknowledgments and responses. The node will, however, continue to send acknowledgments and responses with its proper subnet/node information.
- Authentication cannot be used because the Reply to a Challenge is sent using subnet/node addressing regardless of the addressing format of the original message.
- Nodes are no longer protected against receiving their own messages in looping topologies. This must be considered when designing the application. For example, if a node sends out a network variable update, and it also has an input NETWORK VARIABLE defined with the same network variable selector, its input network variable will get updated if the message is reflected back, which may not be the intention.

debugged and the custom ext step will normally be NEURON 3150 CHIP and different programming issued separately in the

50 CHIP Memory

1 node will always have off-chip EEPROM and in the ROM. The combination of the off-EEPROM. (Refer to *Programmer's Guide*, for application code in the

The off-chip memory for a NEURON 3150-based node is programmed in a PROM programmer using the NEURON ROM Image (.NRI extension) and NEURON EEPROM Image (.NEI extension) exported files. Refer to *Exporting and Importing*, earlier in this chapter, for more information on these files. The NEURON ROM Image (.NRI) file must always be exported and all PROMs must be programmed (whether they contain the firmware or application code) before placing them in the node. The NEURON ROM Image (.NRI) file always includes any application code linked to the ROM area, even if the node being created will come up applicationless, since there is no other way of getting that code there. In the case of an applicationless node, the application code in ROM will not be executed until the node is loaded.

It is also possible to define sections of application code that will reside in EEPROM or non-volatile RAM. This code is contained in the NEURON EEPROM Image (.NEI) file. If your node will come up applicationless, you do not need to program this memory externally. This will be done when the application is loaded. If, however, your node will come up unconfigured or configured, you must program this memory before installation, just like the ROM, since the application must be completely present when the node is powered-up.

The external EEPROM and RAM memory areas will each have a 16 bit CRC value calculated over any application code or data that resides in the area. These values are kept in the respective memory areas, as well as in on-chip EEPROM. Whenever the NEURON CHIP is reset, on-chip and off-chip values are compared, and if there is a mismatch, the node will become applicationless. If the node goes through its automatic initialization of on-chip EEPROM (see below), this check will follow that process, and will override the firmware state selection, if the CRCs do not match.

The one remaining memory area that must be programmed is the on-chip EEPROM. For a NEURON 3150 CHIP there is a data block in the system ROM area (first 16K) that contains a copy of some or all of the on-chip EEPROM memory. How complete this copy is depends on which firmware state is selected when the NEURON ROM Image (.NRI) file is exported. If the state chosen is applicationless, the area will contain only the communication parameters defined for the node. If unconfigured is chosen, the area will also contain application code and data. If the configured state is chosen, this area will contain a complete copy of on-chip EEPROM, including network configuration. When a NEURON 3150 CHIP is powered up and the firmware determines that EEPROM should be initialized (see below), the data from this area will be copied to on-chip EEPROM, and the appropriate firmware state will be set. If the firmware state is applicationless or unconfigured, the remaining EEPROM data must then be loaded over the network. If the firmware state is configured, the chip will be fully programmed at this point.

This special area in ROM will be used to initialize the on-chip EEPROM of a NEURON 3150 CHIP only when the chip is powered up and the firmware detects that EEPROM has not yet been initialized by the current ROM. To accomplish this, there is a special value, or "boot ID," placed in the NEURON ROM Image (.NRI) file when it is exported. This 16 bit value changes each time a NEURON ROM Image (.NRI) file is exported. On power-up, the firmware compares the boot ID in ROM with a location in the on-chip EEPROM. If they don't match, the firmware will initialize the on-chip EEPROM from the special ROM area. It also copies the boot ID to EEPROM, so the initialization will not happen again until the ROM is changed.

Since the boot ID changes each time an NEURON ROM Image (.NRI) file is exported, exporting and creating a new PROM will always result in the EEPROM initialization taking place, even if no changes have been made to the application or configuration. While a node normally only does this initialization once for a given ROM, it is possible to force this process to occur again with the same ROM by resetting the NEURON 3150 CHIP to the blank state (the initial state of EEPROM on a newly manufactured NEURON CHIP) using a special NEURON CHIP firmware image. This image is shipped with the LONBUILDER software in a file named EEBLANK.NRI, and is located in the LONBUILDER system directory. It is shipped in Intel Hex format, but may be converted to Motorola S-record format using the utility HEXCNVRT (see Appendix D for more information on this utility). To reset a NEURON 3150 CHIP's state, program this image into a PROM and power up the node with this PROM in place of the normal firmware PROM. After a short period (about ten seconds for nodes running at 10MHz, longer for slower clocks) the service LED will come on, indicating that the chip has been returned to the blank state. The next time any PROM created from an exported NEURON ROM Image (.NRI) file is placed in the node, the on-chip EEPROM will again be initialized from the special data area in the PROM.

Programming NEURON 3120 CHIP Memory

Since a NEURON 3120 CHIP does not support external memory, the only memory to program is on-chip EEPROM, and this must be programmed over the network.

A blank NEURON 3120 CHIP comes up with its network interface initialized to 1.25 Mbps differential mode (twisted-pair compatible), and a firmware state of applicationless. If your custom node has a compatible transceiver, you can load all of the application and network configuration over the network, using the LONBUILDER Network Manager or another network management tool.

To pre-program a NEURON 3120 CHIP with an application or network configuration other than the default, you must program it in a NEURON 3120 CHIP programmer. (Refer to the documentation supplied with the particular programmer for information on its use.) This type of programmer uses the exported NEURON EEPROM Image (.NEI) file as its input. When exported for use with a NEURON 3120 CHIP, this file has a special format that is not compatible with an NEURON EEPROM Image (.NEI) file exported for a NEURON 3150 CHIP.

Once a NEURON 3120 CHIP is programmed, it may not be possible to program it again if the new communication parameters are different than the default. At this point the only way to change the node's application or network configuration is to load it over a network compatible with the node's new communication parameters.

Installing Custom Nodes

Installation of a custom node in a network involves the identification of the node by a network management tool using the node's NEURON ID. The LONBUILDER Network Manager gets this ID from the service pin message issued when the node's service pin is activated. This means that the node must be able to communicate over the network, which implies that it must have a working transceiver and appropriately configured communication parameters. Communication parameters are set during the programming step. In addition, any routers or bridges between the node and the network manager must be installed and loaded. See *Installing Application Node Target Hardware* in Chapter 6 for detailed instructions on installing.

It is possible to download new communication parameters to a node during the LONBUILDER Network Manager's installation procedure. Transceiver changes, such as bit rate or transceiver type, will require corresponding changes to the node's transceiver hardware.

Changing other parameters, such as minimum clock rate or the custom transceiver timing values, may require all the nodes on that channel to be re-installed with the new parameters in order to communicate. Still other changes, such as the node's priority or hysteresis and filter values, may not require any other actions. Any new communication parameters loaded during installation will not take effect until the node has been reset.

Installing communication parameters also installs the NEURON CHIP input clock as defined in the Hardware Properties screen. If this value is incorrect, the node will not function when reset. If you are using a NEURON 3150 CHIP, you may recover from this error by using a ROM containing the EEBLANK image, as described under *Programming NEURON 3150 CHIP Memory*. If you are using a NEURON 3120 CHIP, the only way to recover from this error is to change the input clock on the node hardware to match the Hardware Properties screen definition.

Loading Custom Nodes

Loading a custom node is required if the node has been created using either the applicationless or the unconfigured state, or if you wish to make changes to the application or configuration without reprogramming the node's memory. (Refer to the section on *Loading Built Nodes and Routers*, earlier in this chapter, for detailed instructions on loading.) A node created in the configured state does not need to be loaded for it to be fully functional.

Reloading a NEURON 3150 CHIP-based custom node with application changes will require the reprogramming step if any application code will reside in ROM. This is because changes to program linkage may invalidate the code already there. If you are going to load the remainder of the application over the network, you only need to program the ROM. If you want the node to come up in the configured or unconfigured state, you must also reprogram any portion of the application image in external EEPROM or RAM. You should load the node after you have installed the new memory.

Using Multiple Firmware Versions

The LONBUILDER IDE supports the use of current or previous versions of the NEURON CHIP firmware images. Whenever possible, use the current version for your development efforts. However, the LONBUILDER IDE accommodates situations where you have already designed and implemented custom nodes which contains a previous version firmware image, and you must continue to work with that previous version.

The Hardware Properties screen (discussed in Chapter 6) permits selection of the firmware image and version. A value of 0 in the firmware version field indicates to the LONBUILDER IDE that it should use the default version. If you need to use a version other than the default, enter the appropriate version number in this screen.

Modifying Firmware Settings

The LONBUILDER IDE already contains the correct default settings as shipped. However, in certain rare situations, you may wish to modify or add to these settings, thus the following information is provided. The information below is **not** the method for setting an individual node's image to be other than the default. The Hardware Properties screen, as already mentioned, is the appropriate mechanism for overriding the default values for one or a few nodes.

The EEBLANK firmware described earlier in this chapter under Programming Custom Nodes is a special form of the NEURON CHIP firmware. It is stored in the LONBUILDER system directory.

This file should only be modified if you wish to permanently change the default values for all nodes of one or more node types in your LONBUILDER system.

Firmware images are stored in subdirectories which correspond to the images' version numbers. These subdirectories are located in the IMAGES subdirectory which is contained in the LONBUILDER system directory. The image subdirectories are named using the template VERnnn, where 'nnn' is 2-255, corresponding to the valid range of NEURON CHIP firmware image version numbers. Each VERnnn image directory will contain the image files (*.NX, *.NXB, *.SYM, and *.IB) as appropriate for the standard images (BIN3120, EMU3120, SYS3120, EMU3150, and SYS3150). These directories also contain special images for the network manager and protocol analyzer, and they contain the 3120 application library, as well as any other libraries.

The IMAGES directory also contains a file named DEFAULT.VER. This file provides information to the LONBUILDER IDE regarding which version number to use as the default for various images. The contents of the DEFAULT.VER file are the correct settings for your version of the LONBUILDER IDE as shipped.

The DEFAULT.VER file contains zero or more lines of the form:

<image-name> <number>

The image name should not have an extension. The number is one to three ASCII decimal digits, without leading zeros, in the range 2-255. White space is required to separate the number from the preceding image name.

A line with an asterisk in place of the image name is used for images whose names do not explicitly appear in the file. Any lines following the first line with an asterisk have no effect. The following is an example of the DEFAULT.VER file:

BIN3120	2
EMU3120	3
EMU3150	3
*	4

The LONBUILDER Editor may be used to modify the contents of DEFAULT.VER. This may be done from within the LONBUILDER IDE. If you modify the contents of this file, always be sure to use the Install All command from the LONBUILDER Project menu immediately afterward. The LONBUILDER IDE does not automatically detect changes in the default values for installation purposes. (An Automatic Build will detect the changes, however, and nodes will be re-linked and re-loaded as appropriate.)

8

Debugging Nodes

This chapter describes how to debug NEURON C programs running on LONBUILDER NEURON Emulators. The NEURON C Developer's Kit and at least one emulator is required to use the debugger. Skip this chapter if you do not have the NEURON C Developer's Kit or an emulator.

NEURON C Debugger

The NEURON C Debugger is a cross debugger, running on the PC while debugging NEURON C applications running on one to 24 LONBUILDER NEURON Emulators. The NEURON C Debugger provides a full-screen source-level view of the application programs executing on the emulators. With the NEURON C Debugger, you can:

- Set breakpoints.
- Start and stop program execution.
- Reset the emulators.
- Single-step through the programs.
- Evaluate, modify, and watch program variables using the NEURON C symbolic names.
- Display the call stack and change the current context to any function within the call chain.

Debugger Window

The NEURON C Debugger window has two subwindows:

- Command button subwindow. Select the `More` command button to display additional command buttons.
- Source subwindow.

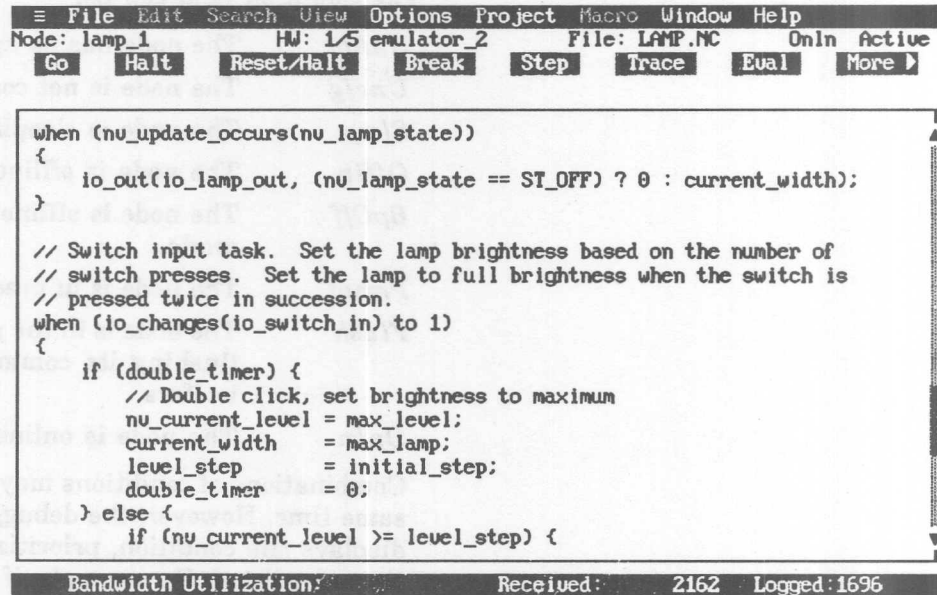


Figure 8-1. Debugger Window

The top border of the Debugger window is always a status line, providing the following information:

Node	Name of the application node currently being debugged (if any).
H W	Development station ID and slot containing the emulator and the target hardware name of the emulator.
File	Name of the NEURON C file being viewed.

Condition	Status of the node independent of the debugger. The condition field can be:	
<i>NoApl</i>		The node has no application.
<i>Uncfg</i>		The node is not configured.
<i>Sleep</i>		The node is sleeping.
<i>Offln</i>		The node is offline.
<i>BpOff</i>		The node is offline in bypass mode.
<i>Prmpt</i>		The node is in preemption mode.
<i>Flush</i>		The node is in the process of flushing its communication buffers.
<i>Onln</i>		The node is online.

Combinations of conditions may exist at the same time. However, the debugger only displays one condition, prioritizing them in the order listed. For example, if a node has an application, is configured, is sleeping, is offline, and is in preemption mode, the debugger displays the condition *sleep*.

Status	Current status of the emulator. The status field can be:	
<i>None</i>		The selected emulator is not loaded with an application image.
<i>Halted</i>		The node is currently halted at a breakpoint or following a single step, or has been halted at a statement boundary through use of the Halt command.
<i>Reset</i>		The node has just been reset. Static initialization has taken place. It is halted at the beginning of the application image.
<i>Active</i>		The node is currently running.

RProt

A read protect error has occurred. (See *Read and Write Protect*, later.)

WProt

A write protect error has occurred. (See *Read and Write Protect*, later.)

Read and Write Protect

The emulator hardware can detect invalid read and invalid write operations. At installation time, the read and write protect map is set up according to the hardware properties defined. (See *Defining Properties*, in Chapter 6.) All unmapped memory is read protected and write protected. All ROM is write protected. RAM, EEPROM, and memory-mapped I/O are not read or write protected.

When a read protect violation occurs, the emulator status is set to RProt and the following message displays:

Node "node name" tried to access non-existent memory
Use the debugger's "Display Stack" command to find nearest source line

When a write protect violation occurs, the emulator status is set to WProt and the following message displays:

Node "node name" tried to write into protected memory
Use the debugger's "Display Stack" command to find nearest source line

In both cases, the node is halted where the violation occurred. The node may not be on a source statement boundary. You can use the Display Stack command to find the nearest source level statement. Then, you can set the context and examine data values as desired. However, it is not possible to proceed from a read or write protect violation. You must reset the node to get it out of this state. (See *Displaying the Current Call Sequence* and *Changing the Current Context*, later, for more information.)

Debugger Commands

This section summarizes the command buttons and accelerator keys for the debugger. The accelerator keys work when either the source subwindow or the command button subwindow is active. To activate the source subwindow:

- 1 Press *Esc* to close any open pull-down menu.
- 2 Press *Esc* to move the pointer from the main menu bar into the debugger window.
- 3 Use the TAB key to toggle the pointer between the command button and the source subwindow, or use the mouse to move the pointer onto the source subwindow and click the mouse button.

Definition

Breakpoint toggle on current line
Clear error log
Display the call stack
Evaluate (and modify) a variable
Go (start or continue execution)
Halt at next statement boundary
Display log for error messages
Reset the node
Single step (step over functions)
Trace step (step into functions)
Move in text subwindow to where
the program halted

Accelerator Key

B
C
D
E
G
H
L
R
S
T
W

Button

Break
Clear Log
Display Stack
Eval
Go
Halt
Log Display
Reset
Step
Trace
Where

Debugger File Markers

The position in a file is set automatically when an application hits a breakpoint, or when you single step, halt, or reset a node. The debugger uses several types of markers to indicate its position within the file, including:

Blinking underscore	Indicates position of the text cursor.
Solid rectangle	Indicates the position of the mouse pointer. You can move the text cursor by moving the mouse pointer to the desired new position of the text cursor and clicking the mouse button.
Triangle cursor	Indicates where the program is halted. (The triangle appears in the left-hand margin of the next line of code to be executed.)
Double-arrow cursor	Indicates the current context, if the current context is different from where the program halted. Although the program is not actually halted at this spot, you can examine or change variables within the indicated context. (A detailed explanation of the current context is presented later under <i>Changing the Current Context</i> .)
Highlighted bar	Indicates a breakpoint. (Highlight bar is colored red for the color sets.)

Running and Debugging Programs

After you have loaded an application image on one or more emulators (described in Chapter 7 under *Loading Built Nodes*), you can run and debug the code. The debugger only works with those applications with source code running on emulators. The debugger cannot be used with SBCs, custom nodes, or imported objects.

Running and Debugging Code

To run and debug code on emulators, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the Target HW button or the Node Specs button. A list of existing nodes displays in the object list.
- 4 Select one or more nodes to debug.
- 5 Select the Reset command button to reset the selected nodes and begin execution from the beginning of the program.
- 6 Select the Go command button to start execution on the selected nodes.
- 7 Select a single node to debug.
- 8 Select the Debug command button to open the Debugger window (see figure 8-1, earlier) with the NEURON C source file for the selected node. The node's status is *Active*.

Pressing the More button displays additional command buttons in the subwindow.

Once the Debugger window is open, the Go To command in the Window system menu can be used to switch between emulators.

When a running program hits a breakpoint or halts due to a Step, Halt, or Trace command, you receive the following message:

```
Node name halted in file pathname\filename.nc  
Due to Breakpoint or user command
```

If the Debugger window is open and a breakpoint is hit on the node you are currently debugging, the debugger automatically sets the file and line corresponding to the location of the breakpoint or halt. When you hit a breakpoint within a program or halt the program, the NEURON CHIP timers stop. When you start the program running again, the timers resume.

Halting Running Nodes

From the debugger, to halt a running program, issue a **Halt** command. The program halts at the next statement boundary. The node status is *Halted*.

From outside the debugger, to halt a running program:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the **App Node** button to list the application node object type selectors—**Properties**, **Target HW**, **App Image**, and **Node Specs**.
- 3 Select the **Target HW** button. A list of existing nodes displays in the object list.
- 4 Select as many nodes as you want to halt.
- 5 Select the **Halt** command button to halt execution at the next statement boundary, or select the **Reset/Halt** command button to reset the node and halt at the reset clause.

Pressing the **More** button displays additional command buttons in the subwindow.

Setting Breakpoints

You can set breakpoints in either a running or halted program. To do so, follow these steps:

- 1 Open the debugger as described earlier under *Running and Debugging Code*.
- 2 Move the text cursor to the line at which you want to set a breakpoint.
- 3 Issue the **Break** command to toggle the breakpoint on. A highlighted bar appears on the line to indicate the breakpoint is set.

Running programs halt just before executing the statement on which the breakpoint is set. After a program halts, you can start execution again with the **Go** command, or single step the program with the **Step** and **Trace** commands.

To remove a breakpoint, follow these steps:

- 1 Move the text cursor to the line containing the breakpoint.
- 2 Issue the Break command to toggle the breakpoint off.

Breakpoints can be set only on *statement boundaries*. If you attempt to set a breakpoint at a point other than a statement boundary, you receive the message

Cursor is not on a statement boundary.

Statement boundaries are any of the following:

- Beginning of a statement that terminates in a semicolon
- Beginning of an if-condition clause
- Beginning of a while loop that has a nonconstant loop expression
- Beginning of a for loop initializer expression
- Beginning of a for loop conditional expression
- Beginning of a for loop iteration expression
- While expression of a do loop
- Beginning of a *when* clause
- Beginning of a task (the body of code following a *when* clause)
- Close brace, which terminates a task or a function definition

A breakpoint can only be set at the first statement boundary on a physical line of source code.

Single Stepping Over Functions

When you single step through a program, the debugger stops execution at each statement as it is encountered. This makes single stepping a useful tool for examining the flow of control within program tasks and functions, and for evaluating when clauses as dictated by the scheduler. When a function call is encountered, single stepping steps *over* the function call and halts at the statement immediately *after* it.

Although single stepping steps *over* functions, it stops at *all* breakpoints, even if they are within function calls. If a breakpoint is hit inside a function that is stepped over, a subsequent Go halts at any other breakpoint encountered in the function, or at the statement following the function call, whichever comes first.

To single step through a program, follow these steps:

- 1 If the node status is *Active*, issue the Halt or Reset command.

or
Set a breakpoint on a line that will be executed and wait for the breakpoint to be hit.
- 2 Issue the Step command to execute the statement pointed to by the triangle cursor and halt at the next statement. If the next statement is a function call, execution halts at the first statement immediately after the call.

When you single step through a program, the timers may behave unpredictably.

Trace Stepping Into Functions

When you trace step through a program, the debugger stops execution at each statement as it is encountered. When a function call is encountered, trace stepping steps *into* the function call and halts execution at the first statement *inside* the function call. If a source line contains multiple function calls, the debugger traces through each function in turn.

To trace step through a program, follow these steps:

- 1 Issue the **Halt** or **Reset** command, if the node status is *active*.
- 2 Issue the **Trace** command to execute the statement pointed to by the triangle cursor and halt at the next statement. If the statement is a function call, execution halts at the first statement immediately *inside* it.

Displaying the Current Call Sequence

The current context defines the local variables and parameters of a function (or task), plus any static variables whose names are not identical to the local variables and parameters.

To display the current call sequence, follow these steps:

- 1 If the node status is *Active*, issue the **Halt** command.
- 2 Issue the **Display Stack** command to open the Call Stack dialog box with the call stack displayed and the current context highlighted (see figure 8-2). When not currently executing application code, the call stack displays

System

When executing application code with a task that has not called any functions, the call stack displays

When Clause or Body

When executing application code within a function, the call stack displays

When Clause or Body
FUNC1
FUNC2
etc.

For example, the application in figure 8-2 is halted in the `new_toggle_timer()` function.

Preemption mode occurs when the NEURON CHIP firmware runs out of buffers. It then tries to free buffers and may invoke application code to do so.

When in preemption mode, the call stack displays

When Clause or Body
FUNC1
FUNC2
etc.
System

and might additionally display

When Clause or Body
FUNC3
FUNC4
etc.

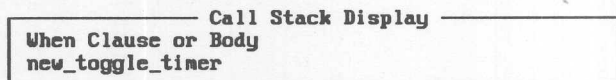


Figure 8-2. Call Stack Dialog Box

Changing the Current Context

By changing the current context, you can examine and modify variables in any function or task body contained in the current call sequence. When you change the current context, the statement at which the debugger is halted does not change.

To change the current context, follow these steps:

- 1 Select one of the unhighlighted contexts in the Call Stack dialog box.
- 2 Press:
Enter or *OK* to change to the selected context.
Esc or *Cancel* to cancel.

When the current context is the same as where the program is currently halted, a single-arrow marker appears.

When you change the current context, the double-arrow marker appears. This marker indicates the local context you've asked the debugger to examine and the line on which it entered the next level of function call, as shown in figure 8-3. Remember that, although you are examining this context, the debugger is actually halted elsewhere in the file.

The screenshot shows a software interface with a menu bar (File, Edit, Search, View, Options, Project, Macro, Window, Help) and a status bar. The status bar displays: Node: sensor_1, HW: 1/2, emulator_1, File: SENSOR.NC, Onln Halted, and buttons for Disp Stack, Log Disp, Clear Log, Where, and More. A code editor window is open, displaying the following C code:

```

when (timer_expires(toggle_timer))
{
    nu_sensor_state = (nu_sensor_state == ST_ON) ? ST_OFF : ST_ON;
    io_out(io_light, (nu_sensor_state == ST_ON) ? LIGHT_ON : LIGHT_OFF);
}

// Switch input task. Update toggle period based on switch on-time.
when (io_update_occurs(io_switch))
{
    new_toggle_timer(io_in(io_switch) / 39); // input & convert to millisecond
}

// Update toggle timer function. Update both the NU and the timer.
void new_toggle_timer(unsigned long new_period)
{
    toggle_timer = new_period;
    nu_toggle_period.milliseconds = (short)(new_period / 10);
}

```

At the bottom of the code editor, a status bar shows: Bandwidth Utilization: Received: 98 Logged: 68.

Figure 8-3. Changing the Current Context

The double-arrow is now inside the task at the point where the function was called. If the task had local variables, you could now look at them and modify them.

Evaluating and Modifying Variables

You can use the debugger to evaluate and modify variable values or the contents of memory given a raw address.

Evaluation can occur when the node is running or halted; modification can occur only when the node is halted.

Evaluation of network variables, timer values, pointer variable values, and static variable addresses has special requirements, which are described later under *Network Variables, Timers, Pointer Variables, and Addresses of Static Variables*.

Trying to directly evaluate or modify memory I/O devices with the debugger will produce incorrect results, since the debugger accesses an emulator's memory over the backplane and will not reach the I/O device. Instead, use the memory browser, described in Chapter 13, to perform this function.

Evaluating a Variable

To evaluate a variable, follow these steps:

- 1 Issue the Eval command to open the Select Data dialog box displaying the string representing the variable name to be evaluated. The default variable name in the evaluation selection window can be chosen as follows:

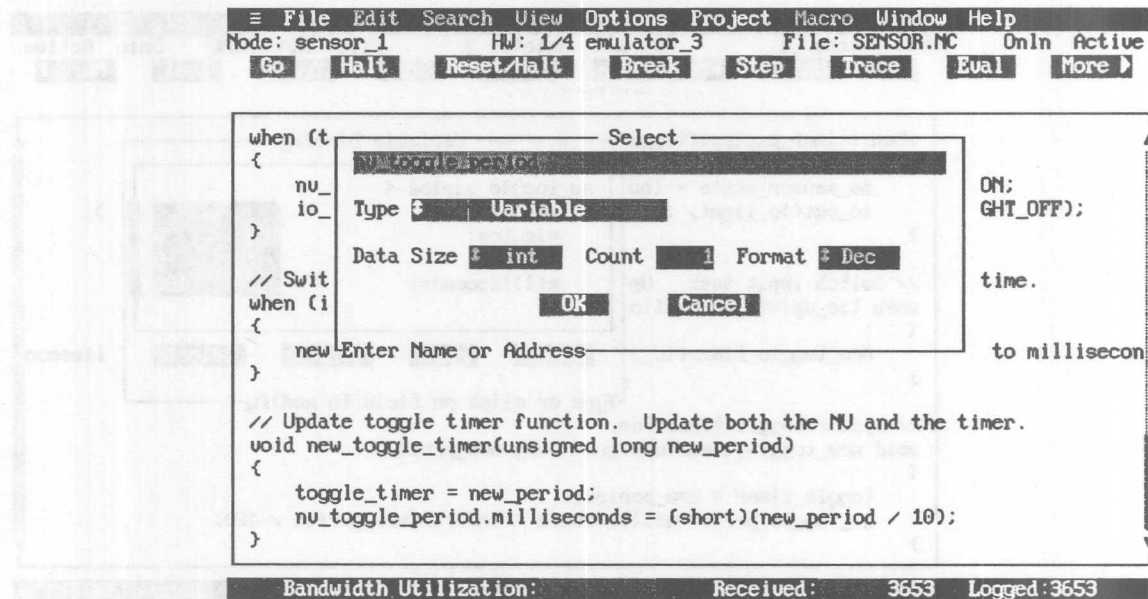
- If the selection occurs because the user chose the Select button on an evaluation screen, the name selected in the previous window will be used.

or

- If the cursor in the display window is sitting on a symbol-like string, the string will be used as the name.

or

- The name that was used by the most recently closed evaluation window will be used again.



- 2 Leave the displayed string, edit it, or replace it with the variable name you want to evaluate.
- 3 Set the Type field to Variable by clicking on it or using the space bar.
- 4 Select the OK command button to open the Variable Display dialog box displaying the current value of the variable (see figure 8-5).

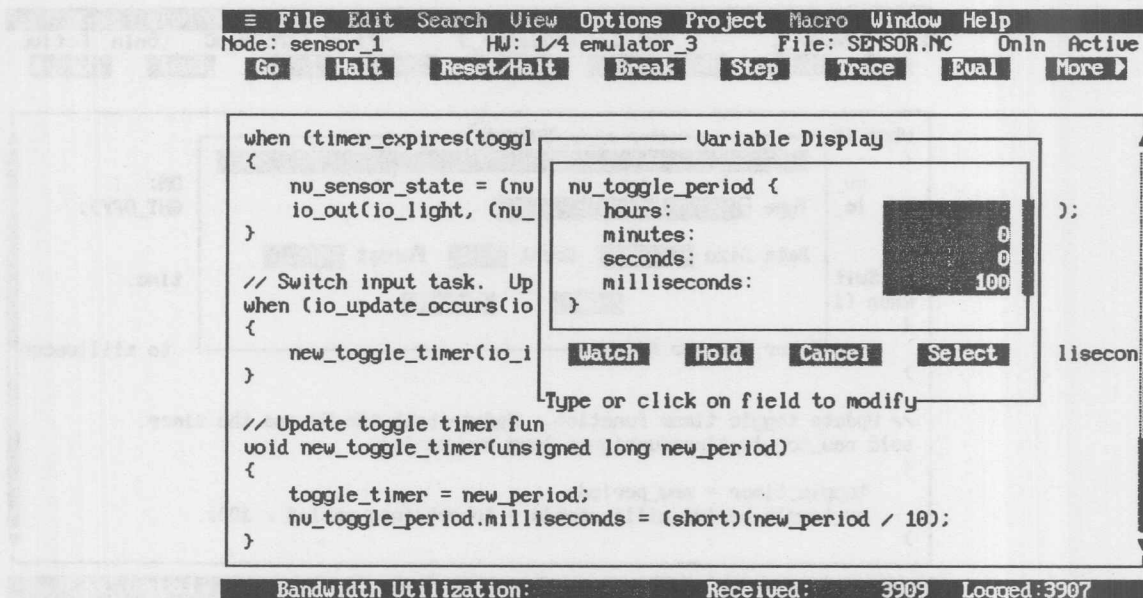


Figure 8-5. Variable Display Dialog Box

5 Select any of the following buttons from the Variable Display dialog box:

A Watch for a local variable or parameter can only be updated when the emulator is halted and the variable is in the current context. Global variables are continuously updated approximately once per second.

Watch

places the window in watch mode. The display is updated every second, and each time the node is halted (whether due to a single step, breakpoint, or halt command). Selecting this button will also transfer control to either the source subwindow or the command subwindow, whichever was previously active. Pressing *Enter* in this window is equivalent to selecting Watch.

Hold	places the window in Display mode. The display will not be updated as the value changes. Selecting Hold also transfers control to either the source subwindow or the command button subwindow, whichever was previously active. The window begins in Display mode.
Cancel	terminates the evaluation window.
Select	terminates the evaluation window, and invokes the Select Data dialog box using the previously evaluated name as the default, allowing you to repeat steps 2 through 4 above.

The debugger supports up to six evaluation windows at the same time. The windows can be moved by dragging them with the mouse. Without a mouse, the windows will stack up (in a cascading fashion) and you will only be able to see one full window at a time.

To activate a given evaluation window either click on the border of the window with the mouse, or use the Next command from the Window menu. The Next window command activates the next evaluation window in a circular manner. Activating a window brings the window to the foreground.

Debugger accelerator functions are not accessible while the evaluation window is active. To enter debugger accelerator keys, control must be transferred to a non-evaluation window. This can be done by clicking on the window using the mouse, or selecting either the Watch or Hold button on the evaluation window.

Displaying an array or structure while the node is active may result in inconsistencies between fields. This is because the application may be modifying the data at the same time the debugger is reading it. To ensure a consistent display of a structure or array, the node should be halted.

Some evaluations can only be made in certain contexts. For example, local variables and parameters can only be displayed when the node is halted and the function containing the variable is in the current context.

If the context is changed, thus invalidating a display of a local variable or a function parameter, the fields of the display will be "greyed out" and protected. If the context is again changed such that the display is valid, the fields will be restored to their normal color and will no longer be protected. A window will be updated only if it is in watch mode and the node is in the proper context for the item to be displayed.

Timers cannot be watched. The watch button will be greyed out and protected when a timer is selected.

Modifying a Variable

The first step in modifying a variable is to evaluate it as outlined in steps 1 through 4 of the section titled *Evaluating a Variable*, earlier. Then, follow these steps to complete the procedure:

- 1 Select by clicking on the displayed value, or type into this field to open the Modify Variable dialog box (see figure 8-6). (The dialog box contains type information and alternate display types, for example: decimal, hex. An enum type variable is displayed with the appropriate enum identifier, if one exists.)

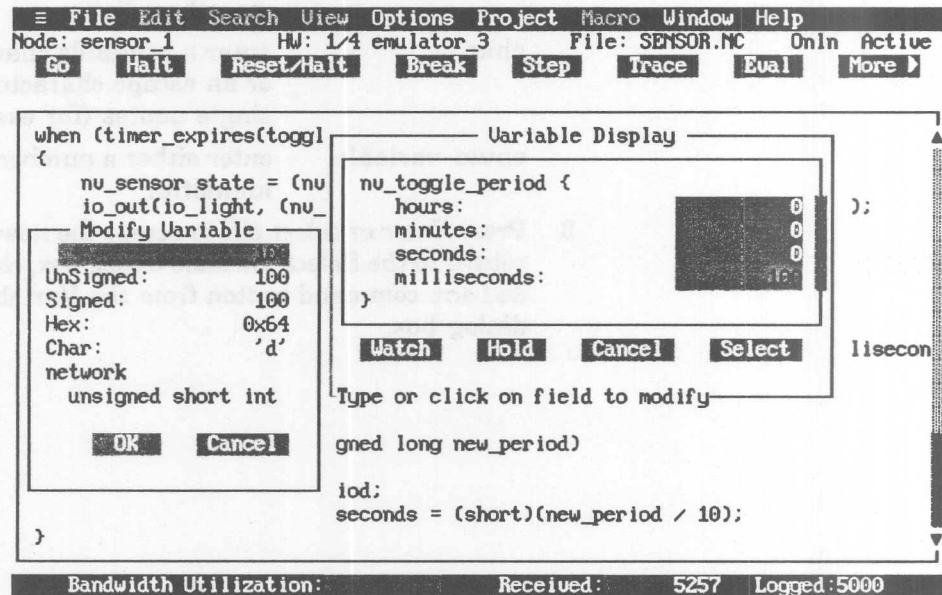


Figure 8-6. Modify Variable Dialog Box

- 2** Enter the new value for the variable. Data entry follows NEURON C syntax. Optionally precede symbolic names with one or more asterisks (*) or with a single ampersand (&). (See *Pointer Variables*, later.) Enter constant values as follows:

decimal	enter one or more decimal digits.
octal	enter 0 followed by one or more octal digits.
binary	enter 0B or 0b followed by one or more binary digits.
hex	enter 0X or 0x followed by one or more hex digits.
char	enter a printable character constant or an escape character enclosed in single quotes (for example: '\n').
enum variable	enter either a number or the enum identifier.

- 3** Press *Enter* or select *OK* to accept the new value. To return to the Select Variable dialog box, choose the *Select* command button from the Variable Display dialog box.

Evaluating and Modifying Raw Memory

To evaluate memory in raw format, follow these steps:

- 1 Issue the Eval command to open the Select Data dialog box. A string representing the *address* of the memory to be displayed will be shown.

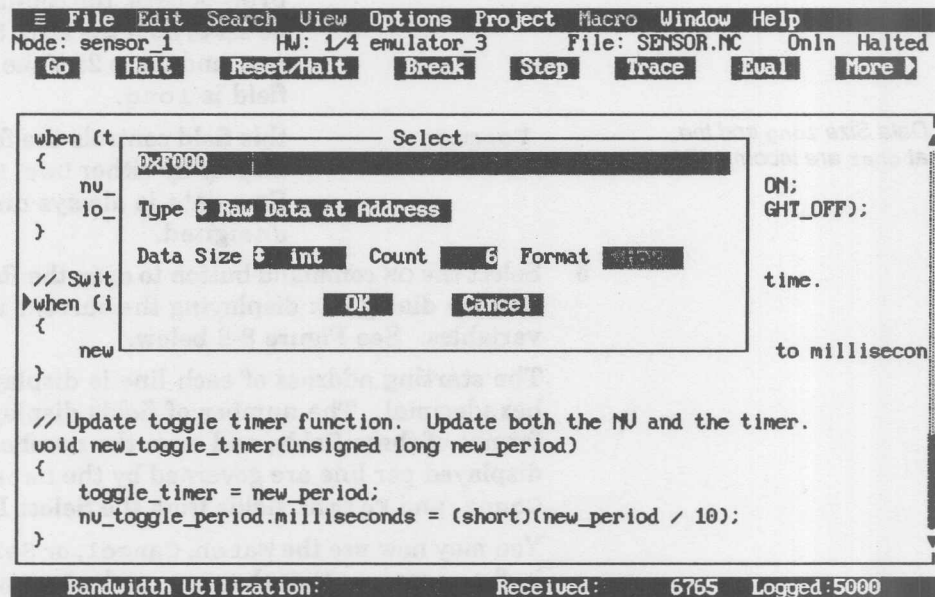


Figure 8-7. Select Data Dialog Box

- 2 Leave the displayed string, edit it, or replace it with the desired address. This value may be entered symbolically. For example, to view the contents of an array *a* as raw data, enter *&a*. To view the data pointed to by a pointer *p* as raw data, enter *p*.
- 3 Set the Type field to Raw Data at Address by clicking on it or using the space bar.

4 Leave or modify the Data Size, At Count, and Format as desired.

Data Size either int or long. Select whether the components of the display should be treated as integers (8-bit) or longs (16-bit).

Count the number of components to display. For example, to display 50 bytes of data, the count field should be set to 50 if the Data Size field is int, and set to 25 if the Data Size field is long.

The Data Size long and the format char are incompatible.

Format this field controls the format of the display by either Dec, Hex, or Char. Raw data is always displayed as unsigned.

5 Select the OK command button to open the Raw Data Display dialog box displaying the current values of the variables. See Figure 8-8 below.

The starting address of each line is displayed in hexadecimal. The number of fields displayed, the format of these fields, and even the number of fields displayed per line are governed by the Data Size, Count, and Format fields from the Select Data screen.

You may now use the Watch, Cancel, or Select buttons in the same way that they are used when evaluating a variable.

If the node is halted, the data fields may be modified in the same way that a variable is modified.

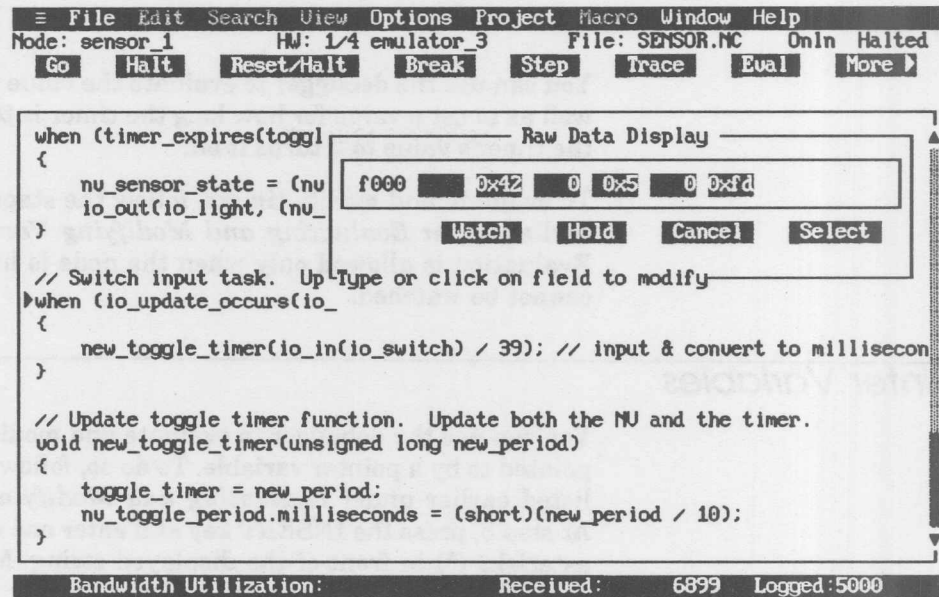


Figure 8-8. Raw Data Display Dialog Box

Network Variables

You can use the debugger to evaluate and modify network variables.

Writing to an output network variable sends an update to all input network variables connected to the output. The update occurs as follows:

- Between the time when the application exits the current task and evaluates the next when clause, or
- When the application calls `post_events()`

When you evaluate an input network variable, the debugger returns the most recent update from the node's program or an output network variable connected to the input.

Timers

You can use the debugger to evaluate the value of a timer, as well as to set a value for how long the timer is to run. Setting the timer's value to 0 turns it off.

To evaluate and modify timers, follow the steps listed earlier under *Evaluating and Modifying Variables*. Evaluation is allowed only when the node is halted. Timers cannot be watched.

Pointer Variables

You can use the debugger to evaluate and modify the value pointed to by a pointer variable. To do so, follow the steps listed earlier under *Evaluating and Modifying Variables*. At step 3, press the INSERT key and enter one or more asterisks (*) in front of the displayed string. Multiple asterisks indicate increased indirection. For example:

```
int i;  
int *p;  
int **pp;  
int ***ppp;  
  
when (reset)  
{  
    i = 3;  
    p = &i;  
    pp = &p;  
    ppp = &pp;  
}
```

Figure 8-9 shows the relationships among the pointer variables. If the program is halted at the ending brace of this code fragment, the values of *p, **pp, and ***ppp would all be 3.

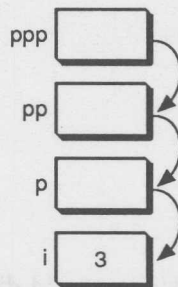


Figure 8-9. Multiple Pointer Variables

When using the * operator, the debugger validates any pointers that it must use to fetch the final value. When the value ***ppp is examined, the debugger validates the contents of ppp, *ppp, and **ppp. For example, if the value of p is 0, the expression *p would result in the following error:

Invalid pointer value

Pointer values cannot point to the return or data stacks, to code area, or to EEPROM. Function pointers cannot be dereferenced.

Addresses of Static Variables

You can use the debugger to evaluate the address of a static or global variable. To do so, follow the steps listed earlier under *Evaluating and Modifying Variables*. At step 2, precede the displayed string with an ampersand (&). The & operator cannot be used on parameters, automatic variables, EEPROM variables, timers, ENUM values, network variables, message tags, or any messaging object (msg_out, msg_in, resp_out, resp_in).

In this example

```
int i;  
int *p;  
int **pp;  
int ***ppp;  
when (reset)  
{  
    i = 3;  
    p = &i;  
    pp = &p;  
    ppp = &pp;  
}
```

the following expressions would display the same value:

ppp

&pp

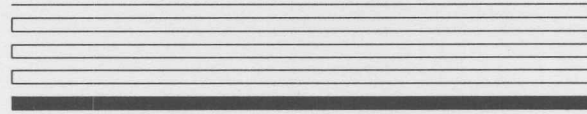
Using the Program Error Log

Whenever your program logs an error or causes the NEURON CHIP firmware to log an error, the words "NEURON Error Log" appear at the bottom of the window. The error code is also placed in the error log.

The debugger checks for errors and detects breakpoints when you are in the Debugger, Navigator, Network Manager, Protocol Analyzer, Statistics, and NV Browser windows. The debugger does not check for errors nor detect breakpoints when you are in DOS, the editor, or the project manager. Errors that occur while you are outside the debugger may not be captured in the error log. If any errors occur that are not captured, this fact is displayed in the error log. Time stamps displayed in the error log indicate when the error was logged by the debugger, not when it occurred.

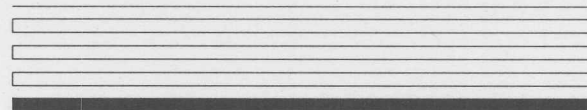
If the error occurred more than 24 hours ago, the date is displayed instead of the time.

There is one error log for up to 24 emulators being debugged. The maximum size of the error log is 25 errors. It is stored in a circular buffer that is reused when it becomes full, with the oldest error being overwritten first. The log is cleared when you select the Clear Log command button. To display the error log, select the Log Display command button. For each message, the log lists the node name and a description of the error. Application program errors can be a decimal number between 1 and 127. System errors are reported using strings.



Part 3

Development Network Installation



9

Defining and Installing Network Management and Router Nodes

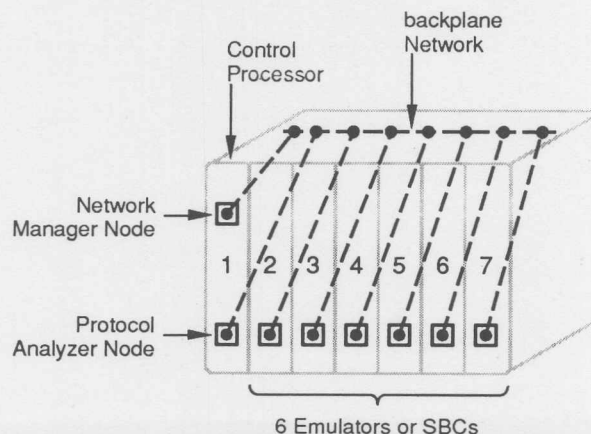
This chapter describes how to define and install the parameters for the network management and router nodes in a development network.

Development Network Integration

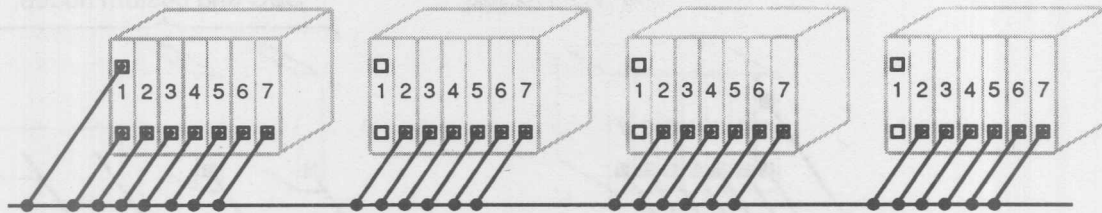
Part 2 of this guide described step 5 of the LONWORKS Application Development Cycle—debug and test individual nodes. Once you have developed individual nodes, you are ready to start step 6—install nodes in a development network and test. The development network serves as a prototype of the production network that will ultimately be created with your nodes. The development network uses the LONBUILDER Network Manager for all network installation and configuration; production networks will most likely use custom network management tools, or may not have any network management tools. See the *NEURON CHIP-based Installation* engineering bulletin for examples of installation without network management tools. See the *LONWORKS Installation Overview* engineering bulletin for examples of different installation scenarios for production networks.

A development network consists of two or more application nodes, a LONBUILDER Network Manager node, and a LONBUILDER Protocol Analyzer node all connected to one or more communications channels. All nodes in a development network must be able to communicate with the network manager node. If multiple channels are used, they must be connected by routers. The following diagrams are examples of development networks:

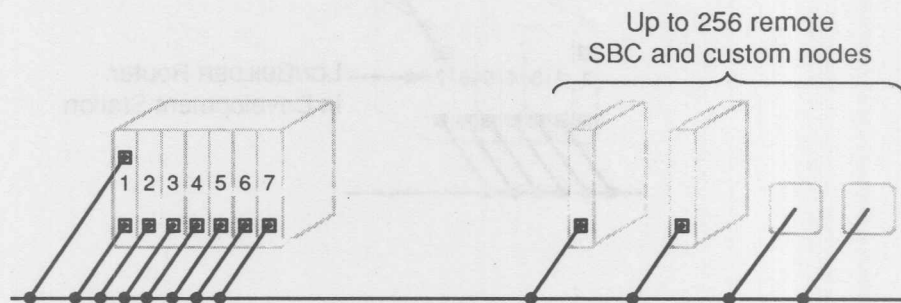
- Up to six emulators and SBCs on the backplane network in the development station.



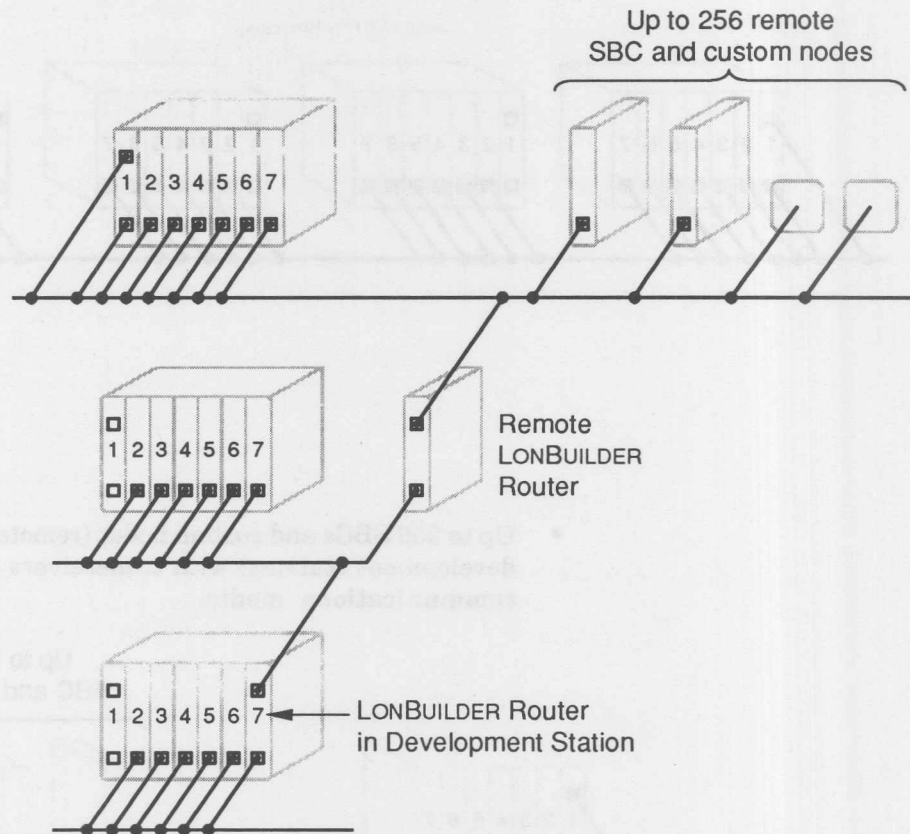
- Up to 24 emulators and SBCs (in up to four development stations) with transceivers on target communications media.



- Up to 256 SBCs and custom nodes (remote from the development stations) with transceivers on target communications media.



- Up to 8 routers to provide routing between multiple communications media (channels).



Network Management Node Target Hardware

Network management nodes support the integration and operation of a LONWORKS network. Two network management nodes are included with the LONBUILDER Developer's Workbench: a network manager node and a protocol analyzer node. The hardware for both of these nodes is contained on the control processor in every development station.

You can connect network management nodes to the backplane network in the development station, or, using optional LONBUILDER or custom transceivers, to any network medium. While you can connect the network management nodes to any channel in your development network, only one protocol analyzer node and one network manager node can be active at any time. The active network manager node controls the nodes on its channel and on any channel connected to it with routers. The active protocol analyzer node analyzes message packets that originate on or are forwarded by a router to the channel to which it is installed. You can re-install the protocol analyzer node as needed to monitor specific channels. The network manager and protocol analyzer nodes can be, but need not be, configured on the same channel.

To change the definition of the network manager and protocol analyzer nodes, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network Mgmt button to list the Network Management object type selectors—Target HW, NV Browser, Packet Log, and Statistics.
- 3 Select the Target HW button to display the Network Manager and Protocol Analyzer hardware definitions in the object list.
- 4 Select either Network Manager or Protocol Analyzer.
- 5 Select the Modify command button to open the Modify window for the selected object. The parameters for the selected item appear in a dialog box.

- 6** Select fields and enter new values as needed. The fields and their defaults are:

Development Station	choose the development station ID in which the node is defined. (The default is 1.)
Channel Name	choose the name of the channel. If the channel is other than default_channel, the channel must be created prior to filling in this field.

- 7** Select a button:

SAVE	modifies the hardware definition with the values in the fields.
CANCEL	returns you to the Navigator window without modifying the definition.

Installing Network Management Nodes Target Hardware

Whenever you change the definition of the network manager or protocol analyzer target hardware, it must be re-installed to ensure that the object database is consistent with the target hardware. You must also re-install target hardware whenever you cycle power on a development station.

To install or re-install the network manager and protocol analyzer target hardware, follow these steps:

- 1** Physically install the control processor hardware as described in the *LONBUILDER Startup and Hardware Guide*.
- 2** Define the network management nodes, as described in this chapter, if you have not already done so.
- 3** Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 4** Select the Network Mgmt button to list the network management object type selectors—Target HW, NV Browser, Packet Log, and Statistics.

- 5 Select the Target HW button to display the Network Manager and Protocol Analyzer hardware definitions in the object list.
- 6 Select the network management nodes you want to install.
- 7 Select the Install command button to receive a message dialog box saying:

Command in Progress

When installation is complete, a message dialog box appears saying:

Command Complete

The project manager can also be used to install or re-install the network manager and protocol analyzer target hardware, as follows:

- 1 Install the control processor hardware as described in the *LONBUILDER Startup and Hardware Guide*.
- 2 Open the Project main menu and select the Automatic Install command. This action installs the hardware in the development station.

Installing Application Nodes, in Chapter 6, describes the different install options on the Project main menu.

Defining Multi-Channel Topologies

A LONWORKS network may consist of more than one channel. Channels are connected by routers. Several types of routers may be defined:

- Learning router, which monitors network traffic to learn the network topology. The router uses the extracted network topology information to selectively route packets between channels.
- Configured router, which uses routing tables based on the network configuration of the nodes to route packets between channels. When selected, the LONBUILDER Network Manager automatically configures the router based on the development network topology.
- Bridge, which forwards all packets between the two connected channels, on the domains in which the router is defined.
- Repeater, which forwards all packets, regardless of domain, between the two connected channels.

Configured and learning routers use the subnet information in LONTALK messages to selectively route messages between channels. Careful configuration of subnets and use of configured and learning routers can reduce overall traffic on some channels. Both configured and learning routers require that if a given subnet appears on one side of it, that subnet cannot appear on the other. If a subnet is to appear on both sides of a router, a bridge or repeater must be used. For configured routers, the network manager configures both subnet and group routing information. Learning routers forward all group messages, but selectively forward group acknowledgements.

The LONTALK protocol requires multi-channel networks to be loop free, tree structured topologies. There is, however, one exception to this rule. A provision has been made allowing "redundant routers". Redundant routers are two or more configured routers connecting the same two channels with identical network configuration. Redundant routers may be used for redundancy or to extend the range of RF networks.

Topology Errors

The LONBUILDER software does not verify that topologies are loop free. If a topology is created with a loop in it, a single message may continually loop through the network.

In addition, various topology errors may be detected by LONBUILDER operations that make use of the network. These are as follows:

- “No logical or physical path found between nodes”

This message may be displayed when doing a build, issuing network management commands from the Navigator, or performing a query or a network peek/poke.

If this message is displayed when a network management command is requested, it means that the target node and the network manager are on different channels and that there is no communication path (established with routers) connecting these channels.

If this message is displayed during binding, it means that two or more of the nodes in the connection that were being built are on different channels, and that there is no communication path connecting these nodes.

When this message is seen, examine the Router Target Hardware configuration to ensure that there is a physical path between the two channels involved. Examine the Router Node Specifications to ensure that the routers that make the physical connection and the nodes involved have at least one domain in common.

- “Redundant routers don't match or not CONFIGURED”

This message may be displayed when doing a build, issuing network management commands from the Navigator, or performing a query or a network peek/poke.

The message indicates that two (or more) routers illegally connect the same pair of channels. Redundant routers are only legal if they are both configured routers, and have identical domain/subnet assignments. If this error occurs during a build, the names of two of the routers involved will be displayed following the message. Either change the Router Target Hardware configuration of one or more of the routers so that they no longer connect the same two channels, or, if they are supposed to be redundant routers, modify their Router Node Specifications to be identical and make sure that they are configured routers.

- “Error: Node <name> cannot be loaded over the network”
or
“Error: Router <name> cannot be loaded over the network”

These messages may be displayed during a build.

The messages indicate that the named node or router (which is not installed in a development station) cannot be loaded over the network because the node or router and the network manager are on different channels, and there is no communication path connecting these channels.

Examine the Router Target Hardware configuration to ensure that there is a physical path between the network manager's channel and the channel of the target node. Examine the Router Node Specifications to ensure that the routers that make the physical connection and the target node have at least one domain in common.

- “Error: Subnet <name> illegally spans channels

See nodes/routers <name> and <name>

Messages on this subnet may not be routed properly”

This message may be displayed when doing a build or a manual load.

It is illegal for a subnet to appear on more than one channel unless those channels are connected solely by bridges or repeaters. This message indicates that this restriction has been violated. The two nodes/routers displayed are configured on this subnet and on different channels that are not connected via bridges or repeaters. This can be fixed in any of the following ways:

- Modify all intervening routers to be either bridges or repeaters.
- Move one of the nodes/routers to another channel such that the subnet does not illegally span channels.
- Put the node or router on a different subnet. Since the subnet field of a node cannot be modified, you may have to create a new node specification (use the copy command) to specify the new subnet. After the new node specification has been created, delete the old one and update any connections that the old node was in.



Figure 9-1. Relationship of Router Hardware Parameters

Defining Router Target Hardware

A router is a node with two NEURON CHIPS that connect two channels. Routers have two types of object definitions in common with application nodes: target hardware, and node specification. Figure 9-1 shows how a router's definitions relate to one another.

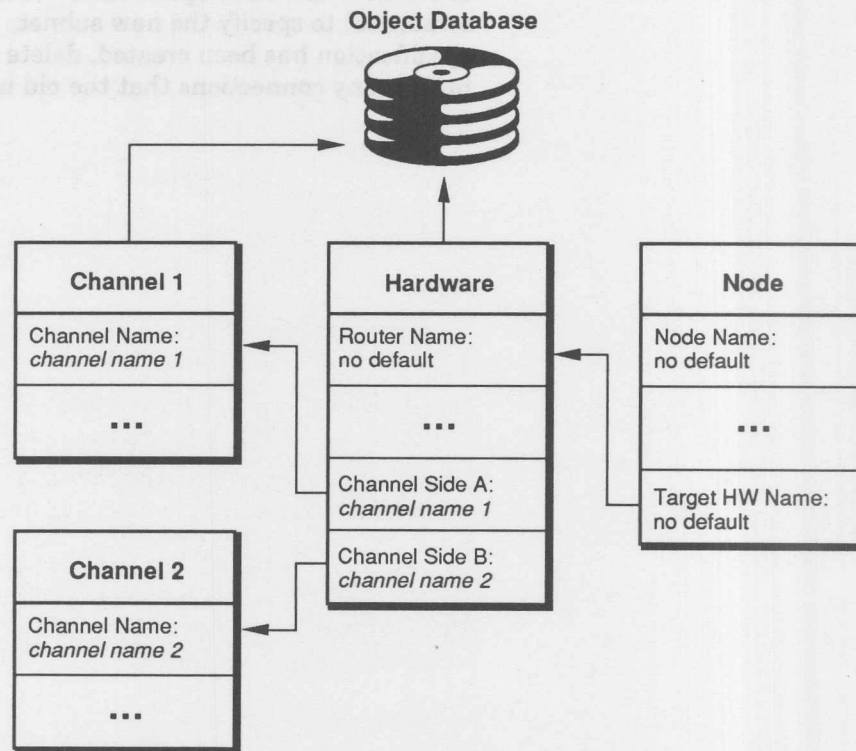


Figure 9-1. Relationship of Router Hardware Parameters

Defining a router's channel definitions is the same as for an application node, as described in Chapter 10, except that two sets of channel parameters must be specified for the two halves. Defining the target hardware parameters for a router is the same as for an application node except that the router objects contain two sets of target hardware parameters, one set for each NEURON CHIP on the router.

The two sets of channel and hardware parameters are specified for side A and side B of the router. Side A and side B are labelled on the LONBUILDER Router. Side A and side B for LONWORKS Routers and LONWORKS Repeaters do not correspond to the Net 1 and Net 2 labels, but are instead determined by the LONBUILDER Network Manager when the router is installed.

This section describes how to define a router's target hardware parameters. A router's node specifications define its network configuration. Chapter 10 includes instructions for defining a router's node specifications.

To create a new target hardware object for a router, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Router button to list the router object type selectors—Target HW and Node Specs.
- 3 Select the Target HW button to display the existing target hardware definitions for routers in the object list.
- 4 Select the Create command button to open the Router Hardware Create window (see figure 9-2).

-or-

Select a router target hardware definition from the object list. Select the Copy command button to open the Router Hardware Create window.

File Edit Search View Options Project Macro Window Help

Router HW Create

Router HW Name: Node Specs: none

HW Type: Location:

Side A		Side B	
Channel Name: <input type="text"/>	<input type="button" value="↓"/>	Channel Name: <input type="text"/>	<input type="button" value="↓"/>
Chan Priority: <input type="text" value="0"/>	Memory: 0 Kb	Chan Priority: <input type="text" value="0"/>	Memory: 0 Kb
Packet Buffers: <input type="text" value="15"/>	Size: <input type="text" value="66"/>	Packet Buffers: <input type="text" value="15"/>	Size: <input type="text" value="66"/>
Clock rate: <input type="text" value="10"/>	MHz	Clock rate: <input type="text" value="10"/>	MHz
Neuron ID: 00-00-00-00-00-00		Neuron ID: 00-00-00-00-00-00	

Name of router hardware:

Bandwidth Utilization: Received: 7837 Logged: 5000

Figure 9-2. Router Hardware Create Window

- 5 Select fields and enter new values as needed. The fields and their defaults are:

Router HW Name

enter a 1- to 16-character unique hardware name. You can use letters, numbers, and underscores in the name. No spaces are allowed. (The name in this field matches the Hardware Name you enter in the node specification, as described under *Defining Router Node Specifications*, in Chapter 10.)

Router HW Type

choose LONBUILDER, LONWORKS Router or LONWORKS Repeater, depending on your equipment type. The LONBUILDER Router is used for development networks. The LONWORKS Routers and LONWORKS Repeaters are used in production networks, but can be installed in a development network using either of these options. The LONBUILDER Router or the LONWORKS Router may be used with any of the four router node types: configured, learning, bridge, or repeater. When using a LONWORKS Repeater, the router type in the corresponding router node spec must be Repeater.

Node Specs

is a read-only field. This field value is assigned when you define the node specifications for this router.

Location

if the hardware type is LONBUILDER, enter the development station ID/slot number of the router. Valid development station IDs are 1, 2, 3, and 4. Valid slot numbers are 2, 3, 4, 5, 6, and 7. (Slot 1 is used by the control processor.) (The default is 1/2.)

If another node already resides in the default ID/slot number, enter an available ID/slot number. If the router will be installed remote from the development station, enter 0/0. If the hardware type is LONWORKS Router or Repeater, this field is read-only and will always be 0/0.

Channel Name

enter the name of the channel for the respective side (A or B) of the node. (You must create the channel definition before you enter its name in this field. See *Defining Channels* in Chapter 10.)

Priority

enter the priority of the router from 0 to 127 for the respective side (A or B). 0 means no priority; otherwise, the lower the number, the higher the priority. (The default is 0. The value in this field must be equal to or less than the value for Num Priorities in the channel definition described under *Defining Channels* in Chapter 10.)

Memory

■ Page 9-16 Correction

The description for the Memory field should read:

Enter the amount of offchip RAM memory installed for buffers (in Kbytes) for the respective side (A or B). For LONBUILDER Routers the value may range from 0 to 6 Kbytes (this memory is always installed), for LONWORKS Routers the value may range from 0 to 26 Kbytes. (The default is 0.) The amount of on-chip memory that is always available for buffers in both types is 1254 bytes. When calculating the number of buffers that will fit in memory, remember to include the four standard buffers that are in addition to the user-specified count. The available memory may not be completely filled since an individual buffer cannot be split between on-chip and off-chip memory. For example, if 20 application buffers are specified, 5 may be allocated to on-chip RAM and 15 may be allocated to off-chip RAM, but none of the individual buffers can be split.

Packet
Buffers

enter the number of non-priority output packet buffers to be used on the respective side (A or B) of this router. Valid packet counts are 2, 3, 5, 7, 11, 15, 23, 31, 47, and 63. (The default is 15.) Enter a smaller value to allow larger packet buffers. These packet buffers are in addition to the two standard input packet buffers and two standard priority output packet buffers. (See Chapter 6 of the *NEURON C Programmer's Guide* for more information.)

Size

enter the size of each input, priority output, and non-priority output packet buffer to be used for the respective side (A or B) of the router. Valid sizes are 66, 82, 114, 146, 210, and 255. (The default is 66.) Larger values may be useful in cases where larger than normal packets need to be handled. (See Chapter 6 of the *NEURON C Programmer's Guide* for more information.)

Clock Rate

■ Page 9-17 Correction

The description for the Clock Rate field should read:

Select the clock rate for the respective side (A or B). Possible rates are 10, 5, 2.5, 1.25, and .625 MHz. (The default is 10 MHz.) LONBUILDER Routers will only operate at 5 or 10 MHz.

NEURON ID

are read-only fields. The values in these fields are assigned when you first install the router.

6 Select buttons as follows:

SAVE	creates a new router target hardware object definition with the values in the fields. After saving, enter new values to create another target hardware object definition, or select CANCEL to return to the Navigator window.
NEXT	displays the values of the next target hardware object selected.
CANCEL	returns you to the Navigator window without creating a new definition.

Installing Router Target Hardware

Whenever you define a new router target hardware object, it must be installed to ensure that the object database is consistent with the router hardware. You must also re-install the router target hardware whenever you change the channel or target hardware definitions of a router.

To install or re-install a LONBUILDER Router in a development station, follow these steps:

- 1 Physically install the router hardware in the backplane, as described in the *LONBUILDER Startup and Hardware Guide*.
- 2 If you have not already done so, define the router's channel and target hardware parameters, as described in this chapter. Set the location to be the development ID/slot number at the router location. Firmware for LONBUILDER routers is installed over the backplane.
- 3 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 4 Select the Router button to list the hardware object selectors—Target HW and Node Specs.
- 5 Select the Target HW button to display the existing router hardware definitions.
- 6 Select the routers you want to install.

- 7 Select the Install command button to receive a message dialog box appears saying:

Command in Progress

When installation is successful, a message dialog box appears saying:

Command Complete

To install or re-install a LONBUILDER Router on a remote network, follow these steps:

- 1 Install the router in a development station as described above.
- 2 Define the router's location parameter to be 0/0. See *Defining Target Hardware*, earlier, for instructions.
- 3 Physically install the router remotely from the development station as described in the *LONBUILDER Startup and Hardware Guide*.

To install or re-install a LONWORKS Router or LONWORKS Repeater, follow these steps:

- 1 Physically install the router as described in the *LONWORKS Router User's Guide*.
- 2 Define the router property, channel, and target hardware parameters, as described in this chapter, if you have not already done so. The location parameter must be 0/0.
- 3 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 4 Select the Router button to list the router object type selectors—Target HW and Node Specs.
- 5 Select the Target HW button to display the existing router target hardware definitions in the object list.
- 6 Select one or more router target hardware objects to install.

■ Page 9-19 Addition

Add the following set of LONBUILDER Router installation procedures following step 3:

To change the communications parameters on a LONBUILDER Router once it has been installed on a remote network, follow these steps:

- 1 Select the router target hardware to install.
- 2 Select the Install button to display the dialog box message:

This hardware has been previously installed.
Use the NEURON ID currently in the database?

- 3 Select Yes to display the dialog box message:

Do you want to install communications parameters?

Follow the instructions for installing new communications parameters on a LONWORKS Router, starting with step 10 on page 9-20 of the *LONBUILDER User's Guide*.

7 Select the Install button. When installing a LONWORKS Router, if you have already installed it at least once, you will first see a dialog box with the message:

This hardware has been previously installed.
Use the NEURON ID currently in the database?

If the NEURON ID has not changed since you last installed the router (you have not swapped the router hardware), you may select Yes to skip the service pin registration and proceed to the optional updating of communications parameters (step 10). If the router hardware has been swapped, or you are not sure which physical router was used when this target hardware was last installed, select No and proceed with the normal service pin registration. In this case, the following message box will be displayed:

To register press service pin. Proceed?

8 Select YES to display the dialog box message:

Waiting for the router's service pin event

9 Press the service pin on the router to be installed. If the network manager does not receive the service pin message, or if you take too long to press the service pin (about 10 seconds), the following message is displayed:

Service pin message not received.

If you get this message, check that a communications path exists from the router to the network manager, and start again from step 7.

If you press the service pin within the allotted time, the following message is displayed:

Do you want to install communications parameters?

10 Select one of the following:

When changing communication parameters, the new parameters do not take effect until the router is reset or power is cycled on the router.

YES

writes the communication parameters to the EEPROM memory on each side of the router target hardware, and displays the message: Command Complete. Installation proceeds to the next selected target hardware. This option should only be used when changing the router's transceiver, channel priority, optional memory, or buffer counts or sizes. After selecting this option, power down the router and then change the transceiver.

The channel bit rate and transceiver type are defined in the Channel Parameters screen. The NEURON CHIP input clock is defined in the Hardware Properties screen. If this value is incorrect, the router will not function after reset.

NO

does not write the communication parameters to the router's EEPROM memory, but does display the message: Command Complete. Installation proceeds to the next selected router. Select this option unless you are changing the transceiver, channel priority, optional memory, or buffer counts or sizes. (This option is the default.)

CANCEL

does not write the communication parameters to the router's EEPROM memory and cancels installation for the remainder of the selected routers.

The Project Manager can be used to install router hardware. To install or re-install routers using the project manager, follow these steps:

- 1 Physically install the router hardware as described in the *LONBUILDER Startup and Hardware Guide* or the *LONWORKS Router User's Guide*.
- 2 Open the Project main menu and select the Automatic Install command. This action installs the routers.

Defining Router Node Specifications

To define the node specifications for a router, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the **Router** button to list the router object type selectors—**Target HW** and **Node Specs**.
- 3 Select the **Node Specs** button to display the existing router node specifications in the object list.
- 4 Select the **Create** command button to open the Router Create window (see figure 9-3).

Router Create

Router name: Router Type:

Side A	Side B
Subnet 1 name: <input type="text" value="default_subnet"/>	Subnet 1 name: <input type="text" value="default_subnet"/>
Subnet 2 name: <input type="text"/>	Subnet 2 name: <input type="text"/>

Auth Key 1: Target HW:

Auth Key 2: Location:

Net Mgt Authenticate:

Router name must be unique

Bandwidth Utilization: Received: 3478 Logged: 3012

Figure 9-3. Router Create Window

5 Select fields and enter new values as needed. The fields and their defaults are:

Router Name

enter a 1- to 16-character unique router definition name. You can use letters, numbers, and underscores in the name. No spaces are allowed.

Type

cycles between router types: Configured, Learning, Bridge, and Repeater. If your router hardware is a LONWORKS Repeater, select Repeater.

Configured routers route packets between the two channels according to routing tables based upon the network configuration.

Learning routers monitor network traffic to learn the network topology. A learning router accumulates knowledge of the network topology to selectively route packets between the two channels.

Bridges Forward all packets between two channels that match one of the router's domains.

Repeaters forward all valid packets between two channels, regardless of domain. Select this type for LONWORKS Repeaters.

Side A Subnet1
Name

enter the name of the subnet in the first domain on channel Side A that contains the router.

Side A Subnet2
Name

enter the name of the subnet in the second domain on channel Side A that contains the router. Leave this field blank if the router is configured in only one domain.

Side B Subnet1 Name	enter the name of the subnet in the first domain on channel Side B that contains the router. Side B Subnet 1 must be in the same domain as Side A Subnet1.
Side B Subnet2 Name	enter the name of the subnet in the second domain on channel Side B that contains the router. Leave this field blank if the router is configured in only one domain. If entered, Side B Subnet2 must be in the same domain as Side A Subnet2.
Target HW Name	enter the name of the router hardware. The name you supply must match the hardware name for the router target hardware. (See <i>Defining Router Target Hardware</i> in this chapter for details.)
Auth Key 1 & Auth Key 2	there are two Authentication keys, one for each domain. These keys are only used if the Network Mgmt Authenticate field is turned on. Each authentication key applies to both sides of the router. Enter the authentication key for each domain in which the router is included. The default is (ff-ff-ff-ff-ff-ff). (Authentication is explained in Chapter 3 of the <i>NEURON C Programmer's Guide</i> .)

Net Mgmt
Authenticate

network management messages that are addressed to the router can optionally be required to use the Authentication keys. Select between YES and NO. When this field is set to YES, the router will authenticate all network management messages that it receives (for example, to change its configuration.) (The default is NO.)

Location

enter an optional 6-character string that identifies the location of the router node in the network. (The Location string is stored in the NEURON CHIP'S EEPROM for maintenance purposes. The default for this field is blank.)

6 Select buttons as follows:

SAVE

creates a new router node specification with the values in the fields. After saving, enter new values to create another router node specification, or select CANCEL to return to the Navigator window.

NEXT

displays the value of the next router object selected.

CANCEL

returns you to the Navigator window without creating a new router node specification.

10

Defining Development Networks

This chapter describes how to define the network parameters of the application nodes—emulators, SBCs, custom nodes, and routers—in a development network.

Loading Nodes in a Development Network

As discussed in Chapter 2, *LONWORKS Application Development*, a node's network image defines its relationship to other nodes and gives it its unique location in the network. The LONTALK protocol propagates messages among network nodes according to an addressing scheme derived from their network images. These addresses are defined using *domain*, *subnet*, and *node* address components, and can be used to address an entire domain, an individual subnet, or an individual node. Another address component, *group*, can be used to address multiple nodes within a domain. The LONTALK address components are defined in the *LONTALK Protocol* engineering bulletin.

There are several options for how the network image is created for a particular node:

- A node may have a static network image created during manufacture. The NEURON ROM image export functions described in Chapter 7 are used to create pre-installed nodes.
- A node may update its own network image using the NEURON C self-installation functions. See the *NEURON CHIP-Based Installation of LONWORKS Networks* engineering bulletin.
- A node may have its network image updated by a network management tool. During development, the LONBUILDER Network Manager is the network management tool for the development network. There are two procedures for creating the network image with the LONBUILDER Network Manager. If an SBC, emulator, or LONBUILDER Router is installed in a development station, then the network image is loaded over the backplane and the network manager is not required. If the node is not installed in a development station, then the network manager sends network management messages to the nodes to configure their network images.

The network manager and protocol analyzer nodes are installed but not loaded because they have no application image and are self contained.

Defining Simple Development Networks

Although your development network may eventually grow to be quite large, you are most likely starting with a single channel of one to six application nodes logically grouped in one domain and one subnet. To make it easy to define simple development networks with only one domain and subnet on a single channel, the node specification definition has a default domain (`default_domain`), a default subnet (`default_subnet`), and a default channel (`default_channel`). Consequently, for simple networks, all you need to do is define the node specification for each node to be installed in the network.

Figure 10-1 shows the relationship of `default_domain`, `default_subnet`, and `default_channel` definitions to the node specification definition, and the relationship of the node specification definition to the target hardware definition.

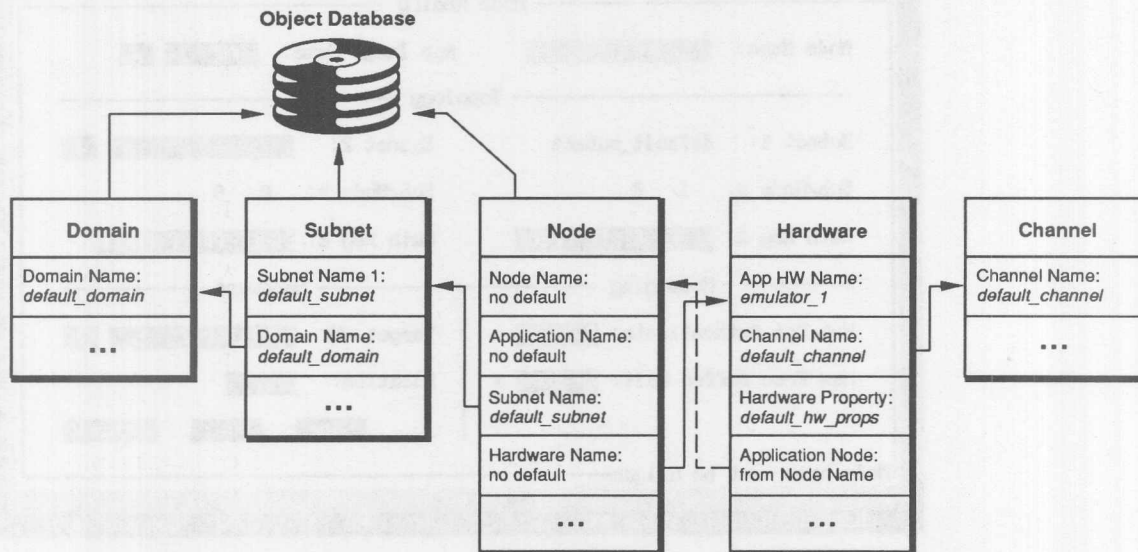


Figure 10-1. Relationship of the Default Network Objects

Defining Node Specifications

Creating node specifications was described in Chapter 6. To change the network configuration of these specifications, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the Node Specs button to display the existing node specifications in the object list.
- 4 Select one or more of the node specifications.
- 5 Select the Modify command button to open the Node Modify window (see figure 10-2).

File Edit Search View Options Project Macro Window Help

Node Modify

Node Name: lamp_1 App Image Name: LAMP ↓

Topology

Subnet 1: default_subnet Subnet 2: [] ↓

Sub/Node #: 1/ 2 Sub/Node #: 0/ 0

Auth Key 1: ff-ff-ff-ff-ff-ff Auth Key 2: ff-ff-ff-ff-ff-ff

Messaging **Hardware**

Net Mgt Authenticate: No Target HU: emulator_2 ↓

Max Free Buffer Wait: 10 s Location: []

[Save] [Next] [Cancel]

Node name must be unique

Bandwidth Utilization: [] Received: 7092 Logged: 5000

Figure 10-2. Node Modify Window

See the LONTALK Protocol engineering bulletin for additional information on these fields.

6 Select fields and enter values as needed. The fields and their defaults are:

Node Name	enter the node name as defined in Chapter 6.
App Image Name	enter the application image name as defined in Chapter 6.
Subnet1 Name	enter the subnet name of the first subnet that contains the node. If you have not defined any subnets, default_subnet, displays in this field. Subnets cannot span configured or learning routers.
Subnet/Node Number 1	the subnet number and the node number of this node within the domain in which subnet 1 resides (read only field).
Subnet2 Name	enter the subnet name of the second subnet that contains the node. Leave this field blank if the node is configured in only one domain. Subnets cannot span configured or learning routers.
Subnet/Node Number 2	the subnet number and the node number of this node within the domain in which subnet 2 resides (read only field).

Node Authentication Key1	enter the authentication key in hex for the first domain in which the node is included. (The default is ff-ff-ff-ff-ff-ff .) An authentication key is only required if the node is programmed for authenticated transactions. Nodes that communicate using authenticated transactions must have the same authentication key. (Authentication is explained in Chapter 3 of the <i>NEURON C Programmer's Guide</i> .)
Node Authentication Key2	enter the authentication key for the second domain in which the node is included. If the node is included in only one domain, leave this field set to the default (ff-ff-ff-ff-ff-ff).
Net Mgt Authenticate?	choose the authentication protocol (YES or NO) for network management messages. (The default is NO.)
Max Free Buffer Wait	choose the maximum wait for a free buffer when sending a message. The choices are 2, 4, 6, 8, 10, 12, or 14 seconds, and infinite wait. (The default is 10 s.)

HW Name

enter the target hardware name as defined in Chapter 6.

Location

enter an optional 6-character string that identifies the location of the node in the network. (The Location string is stored in the NEURON CHIP'S EEPROM for use by a network management tool. The default for this field is blank.)

7 Select buttons as follows:

SAVE

modifies the node specification with the values in the fields. After saving, enter new values to modify another node specification, or select CANCEL to return to the Navigator window.

NEXT

displays the values of the next node object selected.

CANCEL

returns you to the Navigator window without modifying the node specification.

Defining Large Development Networks

You can expand a development network to include up to 24 emulators, 256 remote nodes, and 8 routers. Each node must be included in one domain, and can be included in two; each node must be included in one subnet for each domain in which it is included.

To define the network objects of the nodes in a large development network:

- Define the domain(s) for the nodes. (See *Defining Domains* in this section.)
- Define the subnet(s) for the nodes. (See *Defining Subnets* in this section.) Subnets cannot span configured or learning routers.
- Define the channel(s) for the nodes. (See *Defining Channels* in this section.)
- Define the node specifications for each application node in the network. (See *Defining Node Specifications*, earlier.)
- Define the router definition for any routers used in the network. (See *Defining Router Node Specifications*, later.)

Network objects are created and modified from the Navigator using the Network object type. To create or modify network objects, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network button to list the network object type selectors—Domain, Subnet, Channel, and Connection (see figure 10-3).

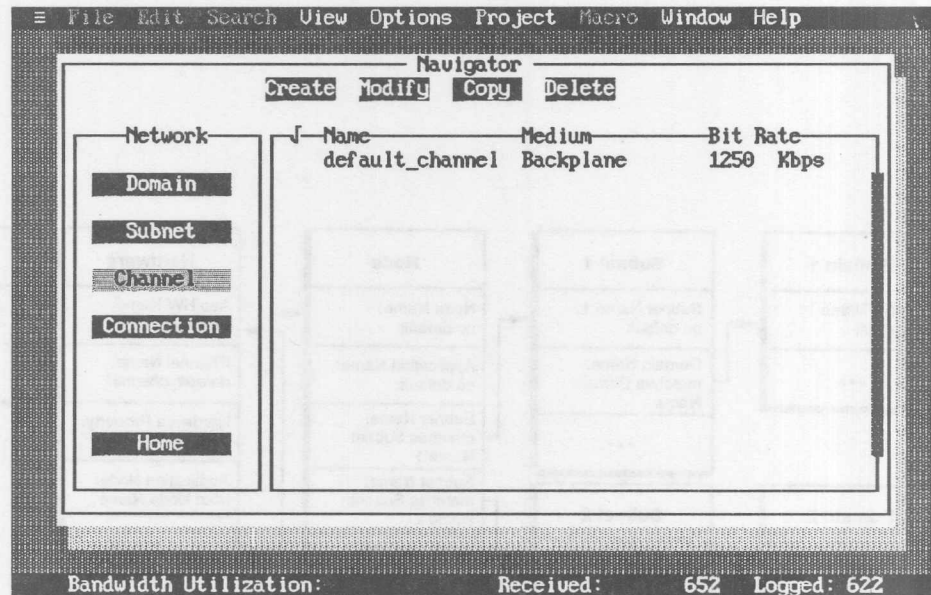


Figure 10-3. Navigator Network Selectors

The Domain, Subnet, Channel, and Connection buttons are used to create and modify object definitions in the object database as described in the following sections.

Figure 10-4 shows the relationship of the domain, subnet, channel, and node specifications for a node, as well as the relationship of the node specifications to the hardware definition.

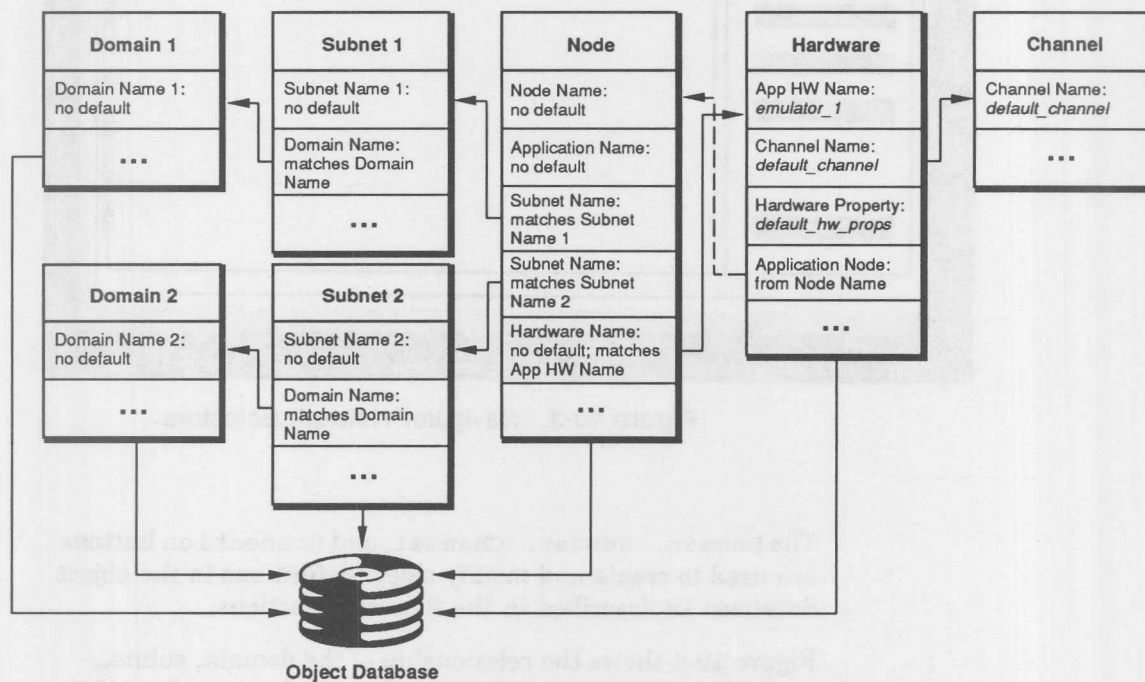


Figure 10-4. Relationship of the Network Definition Objects for a Node in Two Domains

Figure 10-5 shows the relationship of the domain, subnet, channel, and node specifications for a router in two domains, as well as the relationship of the node specifications to the hardware definition.

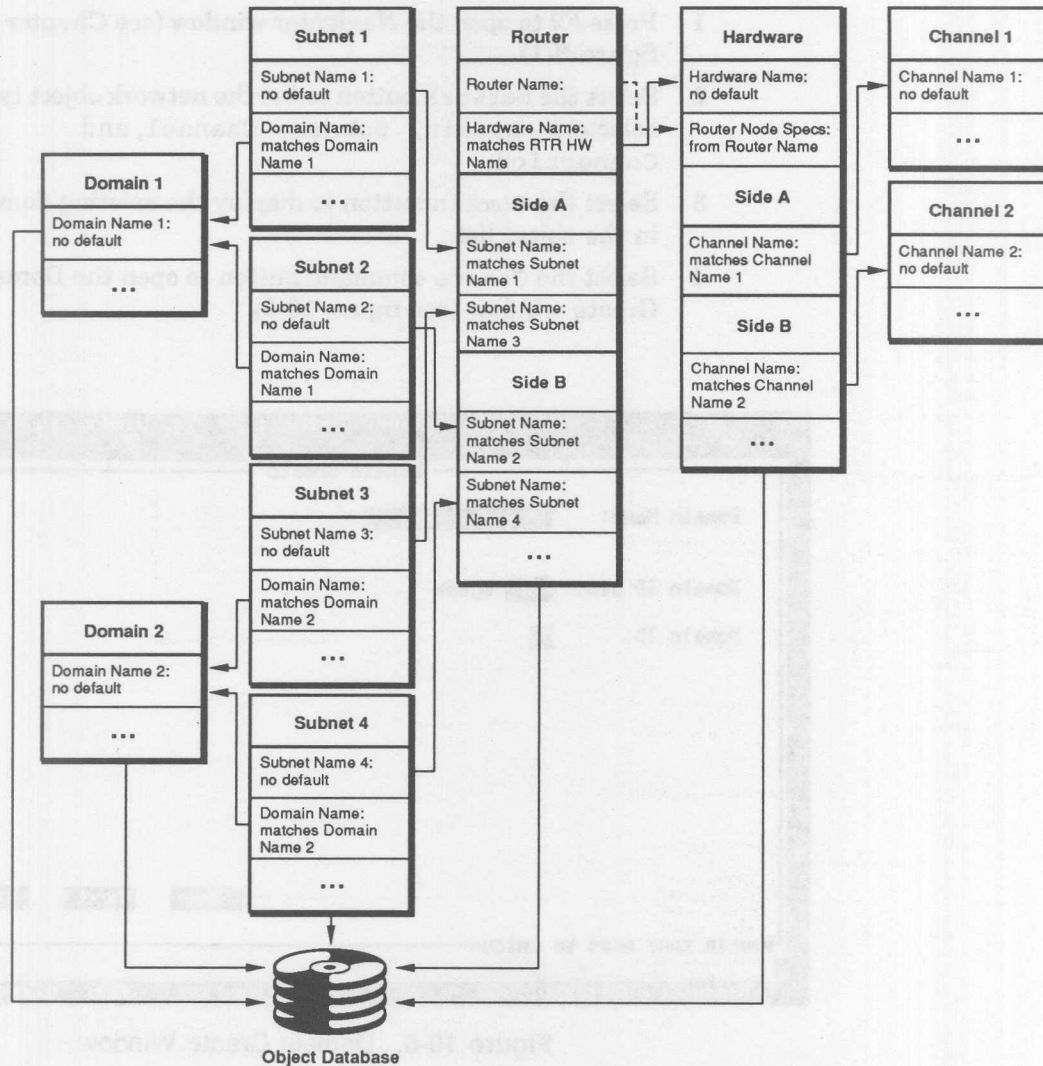


Figure 10-5. Relationship of the Network Definition Objects for a Router in Two Domains

Defining Domains

If you need to create one or more new domains, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the **Network** button to list the network object type selectors—Domain, Subnet, Channel, and Connection.
- 3 Select the **Domain** button to display the existing domains in the object list.
- 4 Select the **Create** command button to open the Domain Create window (see figure 10-6).

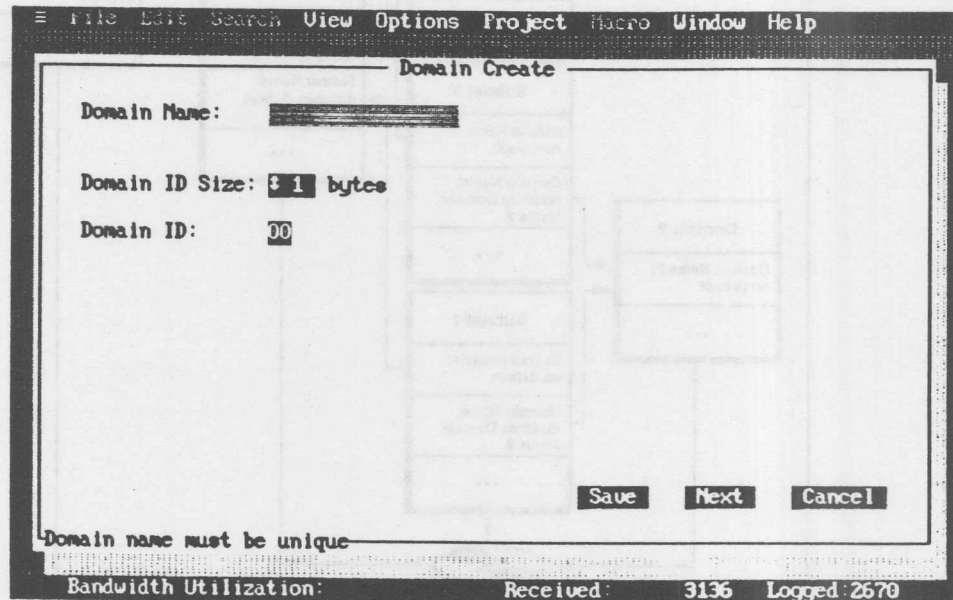


Figure 10-6. Domain Create Window

5 Select fields and enter new values as needed. The fields and their defaults are:

Domain Name

enter a 1- to 16-character unique domain name. You can use letters, numbers, and underscores in the name.

Domain ID Size

choose the length in bytes of the domain ID. Possible sizes are 0, 1, 3, or 6 bytes, where a byte is a 2-digit hex number. Select the smallest domain size possible to maintain the shortest average packet size. If you have a self contained network, you can safely select a 0 length Domain ID Size; if your network includes "open" channels, such as radio frequency and powerline, which could contain traffic from other networks, select a larger Domain ID Size. (The default is 1.)

Domain ID

enter a number that uniquely identifies the domain within the network. If your Domain ID Size is

0 this field is blank.

1 enter a 2-digit Domain ID.

3 enter a 6-digit Domain ID.

6 enter a 12-digit Domain ID.

6 Select buttons as follows:

SAVE

creates a new domain definition with the values in the fields. After saving, enter new values to create another domain definition, or select CANCEL to return to the Navigator window.

NEXT

displays the values of the next domain object selected.

CANCEL

returns you to the Navigator window without creating a new definition.

Defining Subnets

If you need to create one or more new subnets, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network button to list the network object type selectors—Domain, Subnet, Channel, and Connection.
- 3 Select the Subnet button to display the existing subnets in the object list.
- 4 Select the Create command button to open the Subnet Create window (see figure 10-7).

-or-

Select a subnet definition from the object list. Select the Copy command button to open the Subnet Copy window.

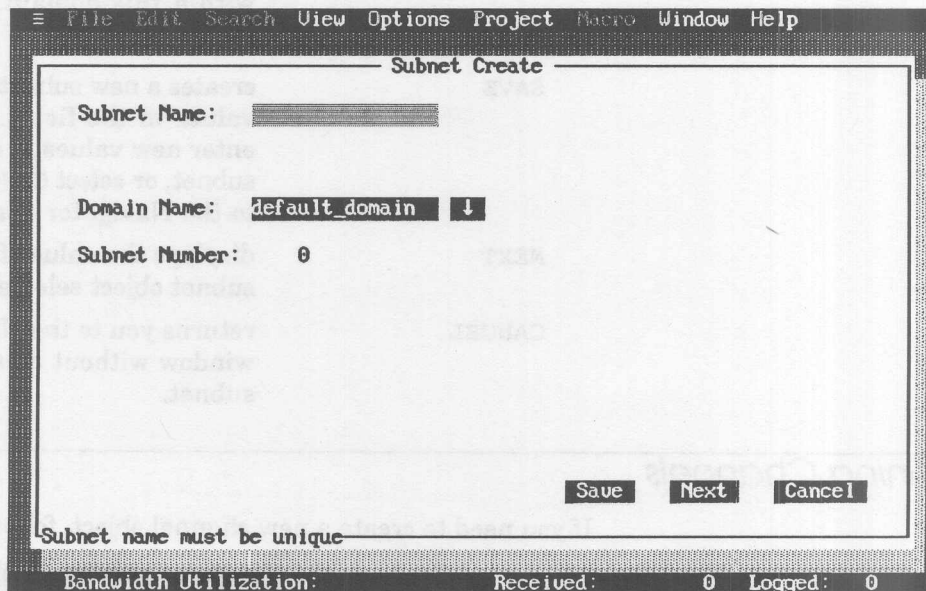


Figure 10-7. Subnet Create Window

- 5 Select fields and enter new values as needed. The fields and their defaults are:

Subnet Name enter a 1- to 16-character unique subnet name. You can use letters, numbers, and underscores in the name.

Domain Name enter the name of the domain within which you want to define this subnet. If you have not defined any domains, the default domain, `default_domain`, displays in this field.

Subnet Number

after the subnet definition is created, this read-only field displays the subnet number within this domain.

6 Select buttons as follows:

SAVE

creates a new subnet with the values in the fields. After saving, enter new values to create another subnet, or select CANCEL to return to the Navigator window.

NEXT

displays the value of the next subnet object selected.

CANCEL

returns you to the Navigator window without creating a new subnet.

Defining Channels

■ Pages 10-16 to 10-20 Replacement

The following text replaces the information for the sub-heading *Defining Channels*:

If you need to create a new channel object, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the **Network** button to list the network object type selectors—Domain, Subnet, Channel, and Connection.
- 3 Select the **Channel** button to display the existing channels in the object list.

LONBUILDER 2.2 Supplement

3-7

- 4 Select the Create command button to open the Channel Create window.

or

Select a channel definition from the object list. Select the Copy command button to open the Channel Copy window.

- 5 Select fields in the main screen and enter values as needed. The fields and their defaults are:

Channel Name enter a 1- to 16-character unique name for the channel. You can use letters, numbers, and underscores in the name.

Std. Xcvt Type select the appropriate LONWORKS standard transceiver type name from the popup selector list. This name should correspond to the type of transceiver you will be using. If you are using a transceiver that does not have a pre-defined type name, select the name CUSTOM. Selecting a type name will cause each of the fields described below to be filled in with a pre-defined value. For any type other than CUSTOM, all those fields (with the exception of Enforce Std Type?) will then become read-only. If the CUSTOM type is selected, the remaining fields may be modified.

NOTE: For specific physical layer recommendations, refer to the LONWORKS Interoperability Guidelines.

Enforce Std Type choose Yes to cause all the remaining fields to be set to their pre-defined values for the selected Standard Transceiver Type name. The fields will then become read-only. Choose No to modify any of the remaining fields. This option will also put an asterisk (*) at the beginning of the transceiver type name to indicate that the remaining values may have been changed. (The default is Yes for standard transceiver types.) If the selected transceiver type is CUSTOM, this field is set to No and is read-only.

Comm Mode choose the NEURON CHIP communication port mode: Single-Ended, Differential, or Special Purpose.

Comm Rate choose the bit rate at which the NEURON CHIP communication port will run: 1250 Kbps, 625 Kbps, 312.5 Kbps, 156.3 Kbps, 78.13 Kbps, 39.06 Kbps, 19.53 Kbps, 9.766 Kbps, 4.883 Kbps, 2.441 Kbps, 1.221 Kbps, or 610 Kbps. For Single-Ended and Differential modes, this rate is the channel bit rate.

For Special Purpose mode this is the rate at which the NEURON CHIP and its transceiver exchange status information and network packets. Since channel activity is reported across this serial interface, it is recommended that it be run at least 8 times faster than the bit rate on the communication medium (the Channel Bit Rate field). It should also run no slower than half the maximum rate supported by the node's input clock. To support this latter constraint across nodes with different clock rates, it is possible to specify a comm rate (interface rate) that is too fast for the minimum input clock. When a node using this slower clock is installed, the comm rate will simply be set to full speed for that clock rate. This only applies to Special Purpose mode.

Regardless of which comm mode is used, the rates below 4.883 Kbps require an input clock rate of less than 5 MHz, and thus cannot be directly supported by the Network Manager, the Protocol Analyzer, or LONBUILDER Routers. In this case, an external router will be required, with one side configured with a slow input clock and transceiver parameters appropriate for the slow communication bit rate. The other side of the router should be configured for a standard LONWORKS channel connected to the LONBUILDER Development Station using a LONBUILDER transceiver. (See the NEURON 3150 and 3120 CHIP Data Book for details on clock rates and bit rates.)

Num Priorities enter the number of priority slots reserved on the channel. You can specify from 0 (for no priorities) to 127. (The default is 0). Nodes must meet certain criteria to be configured on a channel with priority slots. The Input Clock Rate chosen in the hardware properties, and described under *Defining Properties* on page 6-8, affects a node's ability to participate on channels with more than a certain number of priority slots. (See the LONTALK Protocol engineering bulletin for additional information on clock rates and priority.)

Min Clock Rate choose the minimum input clock rate for nodes that will be installed on the channel. Possible minimum input clock rates are 625 MHz, 1.25 MHz, 2.5 MHz, 5 MHz, and 10 MHz. (The default is 10 MHz.)

Avg Packet Size enter the average packet size expected on this channel, a value from 8 to 250, in bytes. (The default is 15.)

Osc Accuracy enter the oscillator accuracy of the nodes on this channel, a value from 1 to 15,000, in parts-per-million. (The default is 200.) If the accuracy varies among the nodes, use the expected worst case. The default value of 200 is suitable for channels where all nodes use crystal oscillators, which are typically rated from 50 to 200 parts-per-million accuracy. If any nodes on a channel use ceramic resonators, then the value should be increased to the worst case for ceramic resonators, which typically range from 1000 to 5000 parts-per-million.

- Osc Wakeup enter the time to wait for a node's oscillator to become stable when waking up from sleep mode, from 0 to 65535, in microseconds. (The default is 0.) This is to ensure that a node woken up from sleep mode by an incoming packet will be able to receive that packet. Note that this time is added to the preamble time for all packets, and thus will decrease performance.
- 6 Select the More button if you wish to modify any of the comm mode specific values or the layer 1 time factors. This will pop up the Comm Mode Specific Parameters window (the Layer 1 Time Factors window is reached by selecting the More button in this new window). Select fields and enter values as needed. The following four fields are displayed only if the Comm Mode field is Single-Ended or Differential. The fields and their defaults are:
- Collision Detect choose Yes to enable collision detection or No to disable. (The default is No.)
- CD Terminate specifies whether the transceiver should terminate the packet immediately after the preamble if a collision is detected before the preamble is completed. The field cycles between Yes and No. Yes is only valid if Collision Detect is enabled. (The default is No.)
- After Preamble
- CD Through specifies whether the transceiver should detect collisions after the code violation at the end of a packet. The field cycles between Yes and No. Yes is only valid if Collision Detect is enabled. (The default is No.)
- Packet End
- Bit Sync allows you to specify the number of transitions it takes to detect whether a real message is being transmitted, as opposed to noise. From the time the NEURON CHIP sees the first transition, until the bit sync is detected, any request to transmit by that node will be denied. Enter a value from 4 to 7, in bits. (The default is 4.)
- Threshold

The following two fields are displayed only if the Comm Mode field is Differential:

- Hysteresis controls the sensitivity of the differential receiver. Enter a value from 0 to 7, encoded. (The default is 0.) Lower values indicate more sensitive receivers, allow you to transmit over greater distances, but increase sensitivity to noise. (Refer to the NEURON 3150 and 3120 CHIP Data Book for the hysteresis values.)
- Filter indicates the level of post hysteresis filtering. Enter a value from 0 to 3, encoded. (The default is 0.) The higher the filtering value, the more filtering that is done. (Refer to the NEURON 3150 and 3120 CHIP Data Book for the filter values.)

The following six fields are displayed only if the Comm Mode field is Special Purpose:

- Channel Bit Rate enter a value from 300 to 312500, specified in bps, for the primary channel bit rate. (Note that the bit rate of the channel is controlled by the transceiver, not the NEURON CHIP. However, the value must be provided here for use in various internal calculations.)
- Alternate Bit Rate enter a value from 300 to 312500, specified in bps, or 0, for the alternate channel bit rate. (Note that the bit rate of the channel is controlled by the transceiver, not the NEURON CHIP. However, the value must be provided here for use in various internal calculations.)
- Backplane The special purpose transceiver mode allows a transceiver to dynamically switch to either an alternate bit rate or an alternate channel. The LONTALK protocol automatically instructs the transceiver to use the alternate frequency or bit rate for the last few retries. If implemented, all transceivers must be able to automatically detect that the transmitter has switched to the alternate channel or bit rate and receive the incoming packet. A value of 0 indicates that no alternate channel is implemented. (The default is 0.)
- Wakeup Pin Dir cycles between Input and Output. NEURON CHIPS have a special low power mode called *sleep mode* which is activated by the application program running in the NEURON CHIP. If the pin direction is configured as Input, when the NEURON CHIP goes into low power mode the transceiver stays active. The transceiver can then wake the NEURON CHIP up for an incoming message by sending a pulse on this pin. A transceiver designer may choose to implement a low power mode for a transceiver. In this case, the pin direction should be configured as Output. When the NEURON CHIP goes into sleep mode it will set the pin to logic low to instruct the transceiver to sleep. With this configuration, the NEURON CHIP will not wake up for an incoming message and must be woken via another method (I/O, timer, or service pin).
- Xcvr Controls cycles between Yes and No. In special purpose mode, the transceiver generates the message preamble. The NEURON CHIP normally controls the preamble length. If your transceiver controls the preamble length independent of the NEURON CHIP, set this field to Yes. (The default is No.)
- Preamble?

General Purpose Data up to seven bytes of user-defined configuration data may be entered to specify the data bytes downloaded to a special purpose mode transceiver after reset. All seven bytes are sequentially downloaded, starting with the left-most byte. However, less than seven configuration registers may be implemented on the transceiver.

Allow Node Override? cycles between Yes and No. Selecting Yes will allow individual nodes on this channel to specify their own unique General Purpose Data (in the Target Hardware screen), overriding the channel's definition. The nodes may also choose to use the channel's definition. Selecting No forces all nodes on the channel to use the channel's definition for General Purpose Data.

7 If you are in the Comm Mode Specific Parameters window, select the More button to switch to the Layer 1 Time Factors window. (Selecting the More button at this level toggles between the two windows.) Select fields and enter values as needed. The fields and their defaults are:

Receive Start Delay indicates the time lapse between the points where the transmitter starts transmitting and the receiver detects the start of transmission. This incorporates delays in both the transmitter and the receiver and should include double the maximum propagation delay for the physical channel. Enter a value from 0.0 to 6553.5 in fractional bit-times. (The default value is 1.0.)

Receive End Delay indicates the amount of time following the end of packet transmission until the receiving NEURON CHIPS have received the end of packet indication. This value accounts for factors such as buffering in the receive side of the transceiver. Enter a value from 0.0 to 6553.5, in fractional bit times. (The default value is 0.0.)

Indeterminate Time indicates the amount of time following the end of the packet where the medium is in an indeterminate state. In particular, during this time, the medium may continue to appear to have activity (e.g. transitions) even though the packet has ended. Enter a value from 0.0 to 6553.5, in fractional bit times. (The default is 0.0.)

Minimum Interpacket indicates the minimum time lapse between the end of one packet and the beginning of the preamble of the next packet. Enter a value from 0.0 to 6553.5, in fractional bit times. (The default value is 0.0.)

Use Raw Data? allows direct entry of sublayer transceiver timing parameters. The LONBUILDER software automatically calculates these values, so the need to use the raw data option should be infrequent. The field cycles between Yes and No. If Yes is selected, the values in the Raw Data field override the time factors. If No is selected, the automatically calculated values are displayed in the Raw Data field. (The default is No.)

Raw Data Clock Rate selects the node clock rate to apply to the Raw Data field. Possible clock rates are .625 MHz, 1.25 MHz, 2.5 MHz, 5 MHz, and 10 MHz. (The default is 10 MHz.) If the Use Raw Data field is No, this value determines the node clock rate used when calculating the data displayed in the Raw Data field. Changing this field causes the data to be recalculated and displayed. (If Use Raw Data is Yes, this value indicates at which clock rate the raw data is valid (raw data must be used at only one clock rate).)

Raw Data The five bytes of communication port data are timing parameters to the NEURON CHIP firmware. In order they are:

Byte 0: preamble length
 Byte 1: packet cycle time
 Byte 2: randomizing slot width
 Byte 3: time between the end of the packet to the earliest start of transmission for the node which transmitted the previous packet
 Byte 4: time between the end of the packet to the earliest start of transmission for nodes which received the previous packet

If the Use Raw Data field is No, this field is read-only, and the values displayed are automatically calculated. If the values cannot be calculated, "?????????" is displayed. (See the NEURON 3150 and 3120 CHIP Data Book for a description of the bits.)

The following fields are displayed only if the Comm Mode field is Differential or Single-Ended, or if the Xcvr Controls Preamble field is No:

Turnaround is the length of time it takes for the transceiver to switch from receiving to transmitting. Enter a value from 0 to 65535, in microseconds. (The default value is 0.)

Missed Preamble indicates how much of the front end of the preamble may not be detected by the receiver, due to start-up anomalies on the channel. Enter a value in fractional bit-times, from 0.0 to 6553.5. (The default value is 0.0.)

The following field is displayed only if the Comm Mode field is Special Purpose, and the Xcvr Controls Preamble field is Yes:

Preamble Length is the length of time required for the transceivers and NEURON CHIPS to achieve bit synchronization on the communications media. (Even though the transceiver is controlling the preamble length in this case, this value is needed for other calculations.) Enter a value from 0.0 to 6553.5 in fractional bit times. (The default value is 1.)

The following field is displayed only if the Comm Mode field is Special Purpose, and the Xcvr Controls Preamble field is No:

Packet Qualification is the amount of preamble that must be seen by the receiver before determining that a valid packet is present. Enter a value in fractional bit-times, from 0.0 to 6553.5. (The default value is 0.0.)

- 8 If you are in the Comm Mode Specific Parameters window or the Layer 1 Time Factors window, select buttons as follows:

OK keeps any changes made in these two windows and returns to the Channel Create window.

CANCEL discards any changes made in these two windows and returns you to the Channel Create window.

- 9 In the Channel Create window, select buttons as follows:

SAVE creates a new channel object with the values in the fields. After saving, enter new values to create another channel object, or select CANCEL to return to the Navigator window.

NEXT displays the value of the next channel object selected.

CANCEL returns you to the Navigator window without saving the currently displayed channel object.

detection or NO to disable (The default is YES for Twisted Pair, and NO for all other transceiver types.)

- 6 Select buttons as follows:

SAVE creates a new channel object with the values in the fields. After saving, enter new values to create another channel object, or select CANCEL to return to the Navigator window.

NEXT displays the value of the next channel object selected.

CANCEL returns you to the Navigator window without creating a new channel object.

Specifying Custom Transceiver Types

There are three custom transceiver type options: Single-Ended, Differential, and Special Purpose. These options are used for development of custom transceivers. The single-ended option is also used for LONWORKS TP-RS485 transceivers. See the *LONWORKS Interoperability Guidelines* for specific physical layer recommendations. When one of these transceiver types is selected in the Channel Create window, an expanded set of parameters is displayed.

To specify a custom transceiver type, follow these steps:

- 1 Press F2 to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network button to list the network object type selectors—Domain, Subnet, Channel, and Connection.
- 3 Select the Channel button to display the existing channels in the object list.
- 4 Select the Create command button to open the Channel Create window (see figure 10-8).

or

Select a channel definition from the object list. Select the Copy command button to open the Channel Copy window.

5 Select the Transceiver Type and enter values as needed. The custom transceiver options and their defaults are:

Bit Rate

for Single-Ended and Differential, the bit rate cycles through 1250 Kbps, 625 Kbps, 312.5 Kbps, 156.3 Kbps, 78.13 Kbps, 39.06 Kbps, 19.53 Kbps, 9.766 Kbps, 4.883 Kbps. For Special Purpose, enter a value from 1200 to 156250, specified in bps. (Note that for the Special Purpose option, this is the channel rate, which is controlled by the transceiver, not the NEURON CHIP. In this case the value will only be used by the protocol analyzer for statistics calculations.)

The following four fields are only displayed when you select the Single-Ended or Differential transceiver types:

Collision Detect

specifies whether the transceiver can detect collisions. The field cycles between YES and NO. The default is NO for all channels except twisted pair, which defaults to YES.

CD Terminate
After Preamble

specifies whether the transceiver should terminate the packet immediately after the preamble if a collision is detected before the preamble is completed. The field cycles between YES and NO. YES is only valid if Collision Detect is turned on. (The default is NO.)

CD Through
Packet End

specifies whether the transceiver should detect collisions after the code violation at the end of a packet. The field cycles between YES and NO. YES is only valid if Collision Detect is turned on. (The default is NO.)

Bit Sync
Threshold

allows you to specify the number of transitions it takes to detect whether a real messages is being transmitted, as opposed to noise. From the time the NEURON CHIP sees the first transition, until the bit sync is detected, any request to transmit by that node will be denied. Enter a value from 4 to 7, in bits. (The default is 4.)

The following two fields are only displayed when you select the Differential transceiver type:

Hysteresis

controls the sensitivity of the differential receiver. Enter a value from 0 to 7, encoded. (The default is 0.) Lower values indicate more sensitive receivers, allow you to transmit over greater distances, but increase sensitivity to noise. (Refer to the *NEURON CHIP Advance Information* document for the hysteresis values.)

Filter

indicates the level of post hysteresis filtering. Enter a value from 0 to 3, encoded. (The default is 0.) The higher the filtering value, the more filtering that is done. (Refer to the *NEURON CHIP Advance Information* document for the filter values.)

The following five fields are displayed only when you select the Special Purpose transceiver type:

Interface Rate

allows you to select the rate at which the NEURON CHIP and its transceiver exchange status information and network packets. Since channel activity is detected across this serial interface, it is recommended that it be run at least 8 times faster than the bit rate on the communication medium. It should also run no slower than half the maximum rate supported by the node's input clock. The NEURON CHIP and associated transceiver exchange status information and data continuously at this rate regardless of any activity on the channel. The interface rate cycles through 1250 Kbps, 625 Kbps, 312.5 Kbps, 156.3 Kbps, 78.13 Kbps, 39.06 Kbps, 19.53 Kbps, 9.766 Kbps, 4.883 Kbps. (The default is 1250 Kbps.)

Xcvr Controls
Preamble

cycles between YES and NO. In special purpose mode, the transceiver generates the preamble message. Either the NEURON CHIP or the transceiver can control the preamble length. The NEURON CHIP normally controls the preamble length. If your transceiver controls the preamble length independent of the NEURON CHIP, set this field to YES. (The default is NO.)

Alternate Rate

enter a value from 1200 to 156250, specified in bps. The special purpose mode allows the transceiver to dynamically switch to either an alternate bit rate or an alternate channel. (Note that this is the bit rate of the channel which is controlled by the transceiver, not the NEURON CHIP. The value will only be used by the protocol analyzer for statistics calculations.) The LONTALK protocol automatically instructs the transceiver to use the alternate frequency or bit rate for the last few retries. If implemented, all transceivers must be able to automatically detect that the transmitter has switched to the alternate channel or bit rate and receive the incoming packet.

General Purpose Data

up to 7 bytes of configuration data may be entered to specify the data bytes downloaded after reset. Data bytes are downloaded in descending order so less than 7 configuration registers may be implemented on the transceiver.

Wakeup Pin Dir

cycles between Input and Output. NEURON CHIPS have a special low power mode called "sleep mode" which is controlled by the application program running in the NEURON CHIP.

The transceiver designer may choose to implement the low power mode for a transceiver. If this pin is configured as Output, the NEURON CHIP will set the pin to logic low to instruct the transceiver to sleep. With this configuration, the NEURON CHIP cannot wakeup for an incoming message. When this pin is configured as Input, the transceiver can wake the NEURON CHIP up for an incoming message. By sending a pulse on this pin, the NEURON CHIP goes into low power mode but the transceiver stays active.

- 6 Select the More button and enter values as needed in the Layer 1 Time Factors window. The options and their defaults are:

Preamble Length

is the length of time required for the transceivers and NEURON CHIPS to achieve bit synchronization on the communications media. Enter a value from 110 to 4600, in microseconds. (The default value is 110.)

Receive Start
Delay

indicates the time lapse between the points where the transmitter starts transmitting, and the receiver detects the start of transmission. This incorporates delays in both the transmitter and the receiver. The time is entered in microseconds. For Single-Ended and Differential enter a value from 1 to 1600. For Special Purpose, enter a value from 1 to 1100. (The default value is 1.)

Receive End
Delay

indicates the amount of time following the end of packet transmission until the receiving NEURON CHIPS have received the end of packet indication. This value accounts for factors such as buffering in the receive side of the transceiver. Enter a value from 0 to 5000, in microseconds. (The default value is 0.)

Indeterminate
Time

indicates the amount of time following the end of the packet where the medium is in an indeterminate state. In particular, during this time, the medium may continue to appear to have activity (e.g. transitions) even though the packet has ended. Enter a value from 0 to 5000, in microseconds. (The default is 0.)

Minimum
Interpacket

indicates the minimum time lapse between the end of one packet and the beginning of the preamble of the next packet. Enter a value from 0 to 10,000, in microseconds. (The default value is 0.)

Use Raw Data?

allows direct entry of sublayer timing parameters. The LONBUILDER software automatically calculates these values, so the need to use the raw data option should be infrequent. The field cycles between YES and NO. If YES is selected, the following field overrides the time factors. (The default value is NO.)

Raw Data

The five bytes of communication port data are timing parameters to the NEURON CHIP firmware. In order they are: the preamble length, the packet cycle time, the randomizing slot width, the time between the end of the packet to the earliest start of transmission for the node which transmitted the previous packet, and the time between the end of the packet to the earliest start of transmission for nodes which received the previous packet. (See the *NEURON CHIP Advance Information* document for a description of the bits.)

7 Select buttons as follows:

OK

to save these changes and to return to the Navigator window.

NEXT

displays the value of the next channel object selected.

CANCEL

returns you to the Channel Create window.

Copying Network Objects

To copy network objects, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the **Network** button to list the network object type selectors—Domain, Subnet, Channel, and Connection.
- 3 Select the **Domain**, **Subnet**, or **Channel** button. Existing definitions for the selected object-type display in the object list.
- 4 Select one or more of the displayed objects.
- 5 Select the **Copy** command button to open the Copy window for the first object selected. All the parameters of the selected definition, except its name, appear in the Copy Window. (The Domain Copy window looks like the window in figure 10-6, the Subnet Copy window looks like the window in figure 10-7, and the Channel Copy window looks like the window in figure 10-8.)
- 6 Select the **Name** field, and enter a unique name for the new domain, subnet, or channel objects you are creating.
- 7 Select buttons as follows:

SAVE	create a new object with the values in the fields. After saving, select NEXT or CANCEL .
NEXT	displays the values of the next object selected.
CANCEL	returns you to the Navigator window without creating a new object.

Modifying Network Objects

To modify network parameters, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the **Network** button to list the network object type selectors—Domain, Subnet, Channel, and Connection.
- 3 Select the **Domain**, **Subnet**, or **Channel** button. Existing definitions for the selected object display in the object list.
- 4 Select one or more of the displayed objects.
- 5 Select the **Modify** command button to open the Modify window for the first object selected. All the parameters of the selected definition appear in the Modify Window. (The Domain Modify window looks like the window in figure 10-6, the Subnet Modify window looks like the window in figure 10-7, and the Channel Modify window looks like the window in figure 10-8.)
- 6 Select the fields you want to modify, and enter new values.
- 7 Select buttons as follows:

SAVE

modifies the object with the values in the fields. After saving, select **NEXT** or **CANCEL**.

NEXT

displays the values of the next object selected.

CANCEL

returns you to the Navigator window without modifying the object.

Deleting Network Objects

You cannot delete network objects that reference other objects. In other words, you must delete all the nodes in a subnet before you can delete the subnet; you must delete all the subnets in a domain before you can delete the domain. To delete network objects, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network button to list the network object type selectors—Domain, Subnet, Channel, and Connection.
- 3 Select the Domain, Subnet, or Channel button. Existing definitions for the selected object display in the object list.
- 4 Select one or more of the displayed objects.
- 5 Select the Delete command button to delete the selected objects.

11

Creating Development Network Connections

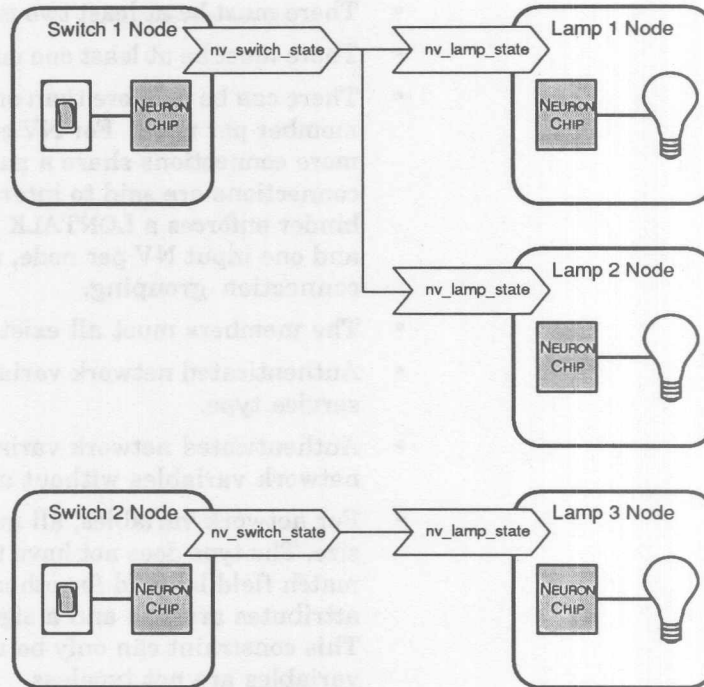
This chapter describes how to connect network variables and message tags and build the network image for the nodes in a development network.

Connections

Connections link nodes' network variable inputs to network variable outputs, and explicit message outputs to explicit message inputs. The connection process causes the generation of address information used by nodes to route data from one node to another. There are two steps to connecting:

- 1 Creating the connection queue. You can do this at any time in the development cycle. This section explains how.
- 2 Invoking a project manager build command *after* the connection queue is created. The project manager invokes the binder to generate the connections and add them to the object database for loading to the nodes. The network manager uses the connection information in the object database to create and send network management messages to configure the nodes' network images.

The connection describes the association of network variable outputs to inputs or the connection of message tags. The example, below, shows a small network of switches and lamps with two connections for the network variables:



These three node/variable name members form a connection for switch1:

<i>node</i>	<i>variable name</i>
switch1	nv_switch_state
lamp1	nv_lamp_state
lamp2	nv_lamp_state

These two node/variable name members form a connection for switch2:

<i>node</i>	<i>variable name</i>
switch2	nv_switch_state
lamp3	nv_lamp_state

The binder enforces the following constraints on connections:

- There must be at least two members.
- There must be at least one output and one input member.
- There can be no more than one input and one output member per node. For NV connections, when two or more connections share a particular input NV, those connections are said to intersect. When this occurs, the binder enforces a LONTALK protocol limit of one output and one input NV per node, for the entire intersecting connection grouping.
- The members must all exist in the connection domain.
- Authenticated network variables cannot use the `unackd` service type.
- Authenticated network variables cannot be connected to network variables without authentication.
- For network variables, all members must be the same size. The type does not have to match for unions, but must match field-by-field for other types. Significant type attributes are size and a signed or unsigned condition. This constraint can only be applied when the network variables are not typeless.

The binder issues a warning message when:

- A polled output network variable is not attached to at least one polling input network variable.
- Both synchronized and nonsynchronized network variables are included in the same connection.

Creating Connections

To create an entry in the connection queue, follow these steps:

- 1 Press **F2** to open the Navigator window (see figure 3-1).
- 2 Select the **Network** button to list the network object type selectors—Domain, Subnet, Channel, and Connection.
- 3 Select the **Connection** button to list the connection object type selectors—Net Variable and Message Tag.
- 4 Select the **Net Variable** button to list the existing network variable connections in the object list, or select the **Message Tag** button to list the message tag connections.
- 5 Select the **Create command** button to open the **Network Variable** or the **Message Tag Connection Create** window. Both windows contain the same fields. Figure 11-1 shows the **Network Variable Connection** window. You use this window to list the members of a connection.

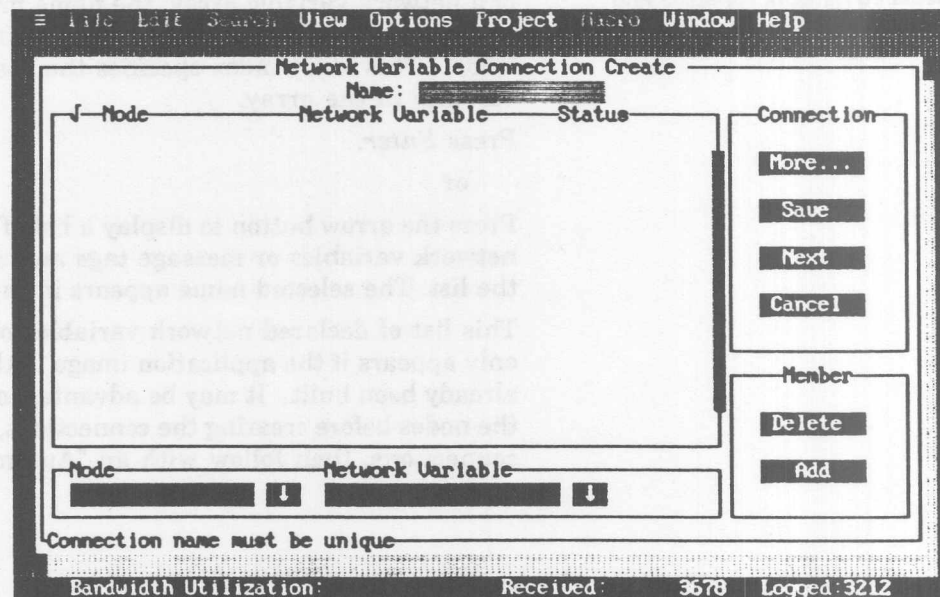


Figure 11-1. Network Variable Connection Create Window

- 6 Select the Name field and enter a name for the connection. The connection name is a unique name from 1- to 16-characters of letters, numbers, and underscores. No spaces are allowed.

To add members to a connection:

- 7 Select the Node field.
- 8 Enter the application node name of a member and press *Enter*.

or

Press the arrow button to display a list of application nodes and select one node from the list. The selected node name appears in the field.

- 9 Select the Network Variable or the Message Tag field.
- 10 Enter a network variable or message tag name of a member. The network variable name is a unique name from 1- to 16-characters of letters, numbers, and underscores. The first character cannot be a number. No spaces are allowed. For members that are elements of a network variable array, the name must be followed by an array index enclosed in brackets, for example "[13]". This array index specifies the position of the variable in the array.

Press *Enter*.

or

Press the arrow button to display a list of declared network variables or message tags and select one from the list. The selected name appears in the field.

This list of declared network variables or message tags only appears if the application image of the node has already been built. It may be advantageous to build all the nodes before creating the connections, then create the connections, then follow with an "Automatic Load".

The arrow button only displays a list of defined nodes if there is enough memory in the LONBUILDER PC to do so.

The arrow button only displays a list of network variables or message tags if there is enough memory in the LONBUILDER PC to do so and if you have run a build command, which compiles and links the application code for all nodes in the project and places their declared network variables and message tags in the object database.

- 11 Select the ADD button to add the network variable or message tag member to the connection. Once added, the member's status is ADDING.
- 12 Repeat steps 7 through 11 until you have finished adding members for the connection.
- 13 To create another connection, select SAVE, select NEXT, and go to step 4; otherwise select SAVE and select CANCEL.

Modifying Connections

To modify existing connections, follow these steps:

- 1 Press *F2* to open the Navigator window (see figure 3-1).
- 2 Select the Network button to list the network object type selectors—Domain, Subnet, Channel, and Connections.
- 3 Select the Connection button to list the connection object type selectors—Net Variable and Message Tag.
- 4 Select the Net Variable button to list the existing network variable connections in the object list, or select the Message Tag button to list the message tag connections.
- 5 Select one or more connections.
- 6 Select the Modify button to open either the Network Variable or the Message Tag Connection modify window for the first connection selected (see figure 11-3). The name of the connection displays in the Name field. The network variable or message tag members of the connection and their status display in the window. Status can be:

ADDING	member is queued for binding (inclusion) in the object database.
CONNECTED	member is connected and the object database is current.
MODIFYING	member is connected, but is queued for changes in the object database.
DELETING	member is connected, but is queued for deletion from the object database.

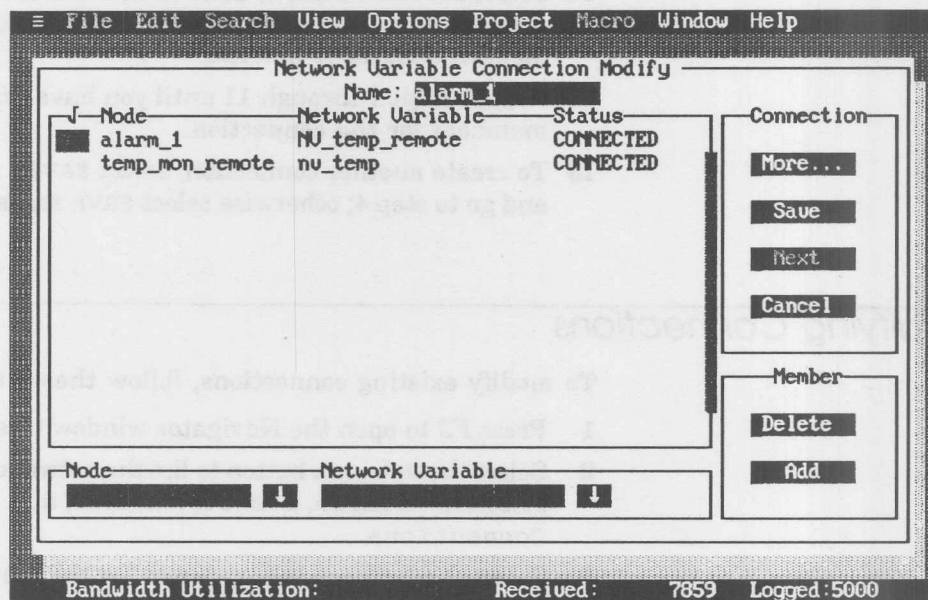


Figure 11-2. Network Variable Connection Modify Window

To add a member, follow the directions under *Creating Connections*, earlier.

To delete a member:

- 7 Select the member from the object list.
- 8 Select the DELETE command button.

To change an incorrectly spelled node, network variable, or message tag name:

- 9 Delete the incorrect network variable or message tag.
- 10 Add the correct network variable or message tag.

To change the name of the connection:

- 11** Select the Connection Name field.
- 12** Type the new name over the existing name.

When you're finished making changes:

- 13** Select the SAVE button.
- 14** Select the NEXT button to bring the next connection you selected to the window or select CANCEL.

Deleting Connections

To delete an entire connection, follow these steps:

- 1** Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2** Select the Network button to list the network object type selectors—Domain, Subnet, Channel, and Connections.
- 3** Select the Connection button to list the connection object type selectors—Net Variable and Message Tag.
- 4** Select the Net Variable button to list the existing network variable connections in the object list, or select the Message Tag button to list the message tag connections.
- 5** Select one or more connections.
- 6** Select the Delete command button to delete the connection.

Setting Parameters for Connections

When writing NEURON C application code, you can specify message parameters for network variable and message tag connections. You can also specify if these parameters can be altered by a network management tool.

To use the network management tool within a development station to alter message parameters for a network variable or message tag connection, follow these steps:

- 1 While in the Connection Create (see figure 11-1, earlier) or the Connection Modify (see figure 11-2, earlier) window, select the MORE button to open either the Network Variable (figure 11-3) or the Message Tag (figure 11-4) message parameter window.

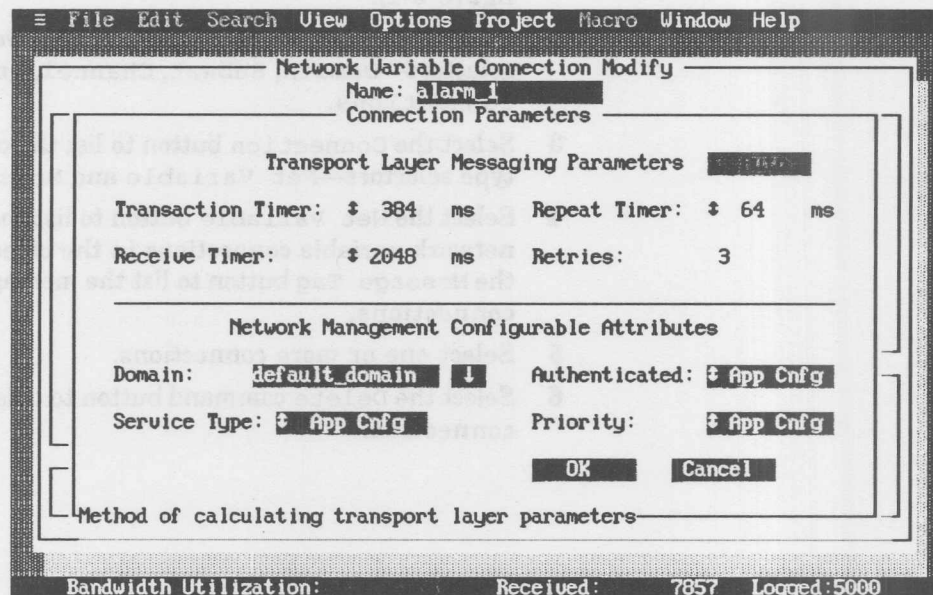


Figure 11-3. Network Variable Connection Parameter Window

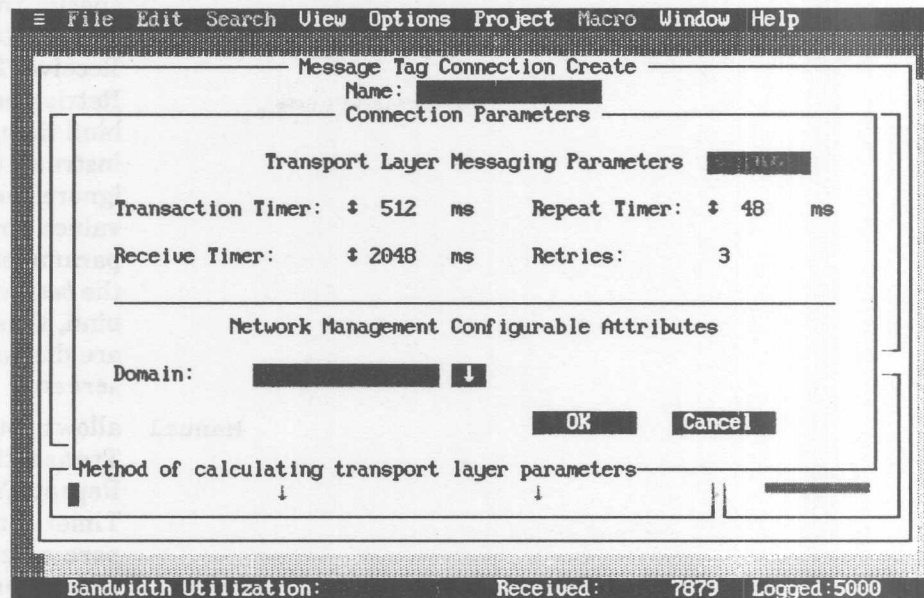
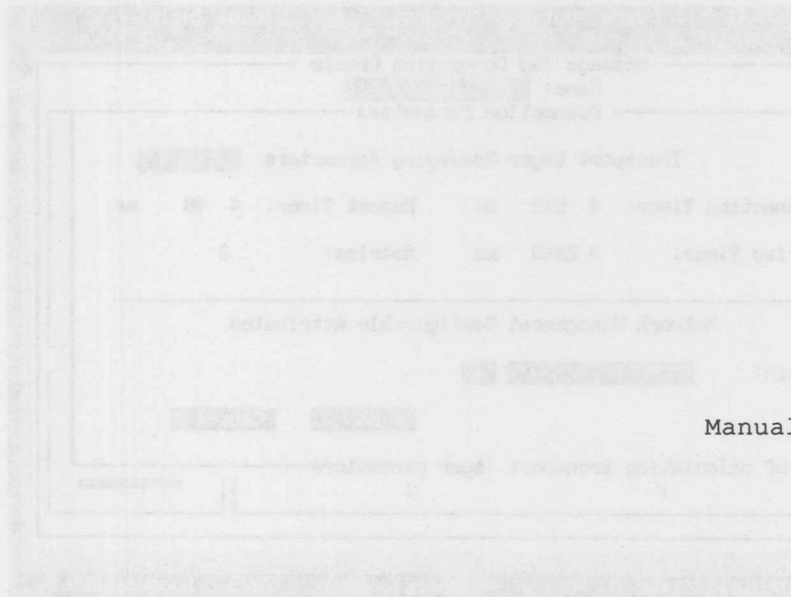


Figure 11-4. Message Tag Connection Parameter Window

- 2 Select fields and enter values as needed. The fields and their defaults are:

Transport Layer Messaging Parameters	choose Auto mode or Manual mode. (The default is Auto .)
--------------------------------------	---



Auto does not allow you to specify the Transaction Timer, Repeat Timer, Receive Timer, and Retries parameters. At bind time, Auto instructs the binder to ignore the specified values for these parameters and choose the *best* value. After the bind, these "best" values are displayed in this screen.

Manual allows you to specify the Transaction Timer, Repeat Timer, Receive Timer, and Retries parameters. At bind time, Manual instructs the binder to use the specified values for these parameters.

The proper settings for transaction and receive timers depend on many factors, some of which are application dependent. When using the Auto mode, the LONBUILDER software will calculate values based on certain assumptions. You may need to adjust the timers to be larger or smaller for any given application. The calculations used for channels using special purpose transceivers are particularly likely to have values which are too small.

Refer to the LONTALK Protocol engineering bulletin and the Optimizing LONTALK Response Time engineering bulletin for information on when to change the defaults in the Transaction Timer, Repeat Timer, Receive Timer, and Retries fields.

Transaction
Timer

enter a transaction timer value. This is the time interval between retries when ackd service is used. Timer choices range from 16 ms to 3072 ms. The default depends on the network topology.

Repeat Timer

enter a repeat timer value. This specifies the time interval between repetitions for an outgoing message when `unackd_rpt` service is used. Timer choices range from 16 ms to 3072 ms. The default depends on the network topology.

Receive Timer enter a receive timer value. When the node receives a multicast (group) message, the receive timer is set to the value specified here. If a message with the same transaction ID is received before the receive timer expires, it is considered to be a retry of the previous message.

The binder automatically sets the receive timer for non-group messages to be the largest of the receive timers specified for non-group (subnet-node) messages. Timer choices range from 128 ms to 24576 ms. The default depends on the network topology.

When manually setting a receive timer, be sure that it is at least as large as $(\text{transaction timer}) * (\text{number of retries} + 1)$. A better formula is $(\text{transaction timer}) * (\text{number of retries} + 2)$.

enter a message retry count. The default is 3. This count does not include the first try.

Retries

Domain

enter a domain name and press *Enter*.

or

press the arrow button to display a list of defined domains and select one from the list. If you have not created any domains, `default_domain` is the only domain name in the list.

The following three fields are only a part of the Network Variable Connection Parameter Window.

Authenticated

choose YES, NO, or `AppCnfg`. (The default is `AppCnfg`, which means the authentication service declared in the NEURON C program is used.)

Service Type

choose `ackd`, `unackd`, `unackd-rpt`, or `AppCnfg`. (The default is `AppCnfg`, which means the service type declared in the NEURON C program is used. Service Type must be `ackd` if authentication is set to YES.)

Priority

choose YES, NO, or `AppCnfg`. (The default is `AppCnfg`, which means the priority declared in the NEURON C program is used. If YES is chosen, the network variable update is transmitted in a priority slot. The node must also be configured to have a priority slot in the Application Hardware window.)

- 3 Select SAVE to save the message parameters as specified; select CANCEL to return to the Network Variable or Message Tag connection window without saving.

Building the Network Image

After defining the connections, you issue a project manager build command to:

- Bind the connections in the database. Binding automatically checks the connections' network variable and message tag types for compatibility, and selects the appropriate network addressing modes to use (unicast or multicast). First, all nodes within a connection are found. Then, for each connection, addresses are assigned to all appropriate nodes to ensure that information flows to and from the right nodes.
- Update the network image on the nodes on the development network.

To issue a build command:

- 1 Open the Project pull-down main menu.
- 2 Select one of the following commands:

Automatic Build	Selecting this command recompiles and relinks the project's NEURON C and include files if needed. The Automatic Build command calculates all dependencies and recompiles files that have changed and/or are dependent on include files that have changed since the previous build.
-----------------	--

If the Stop After Compile project configuration parameter is off, the Automatic Build command invokes the binder to connect message tags and network variables when

- The connection interface files change as a result of the build. This means that the application has changed its interface to other nodes (i.e., NV or message tag declarations have changed.)
- You specify a new or different set of connections.
- You add a new node, or change network parameter information (see Chapter 10, *Defining Development Networks*).

If the Load After Build project configuration parameter is on, the Automatic Build command loads the application images after connecting. It also updates any network images that have changed as a result of any new or modified connections. If the Start after Load project configuration parameter is on, the Automatic Build command starts all newly loaded and configured nodes.

Build All

Selecting this command recompiles and relinks all program files. If the Stop After Compile project configuration parameter is off, the Build All command invokes the binder to reconnect all network variables and message tags. If the Load After Build project configuration parameter is on, the Build All command loads the application images after connecting. It also updates any network images that have changed as a result of any new or modified connections.

Automatic Load

Selecting this command loads existing application images to the project's nodes if the node's application image is out of date. If you select this command, any necessary compiles, links, or binds will be done prior to loading. This command is identical to an Automatic Build with the Load After Build option set.

If the project manager encounters any connection errors, the user should proceed to the Connection Modify window to fix them. See *Modifying Connections*, earlier, for instructions on changing connections.

Node Configuration Reports

A Node Configuration Report contains information regarding a particular node's application and network images. This report is a useful tool when debugging nodes, building systems, or binding nodes. The node image information included in this report may be of particular interest to users writing self-installing applications. Refer to the *NEURON C Programmer's Guide* for additional information on this subject.

A Node Configuration Report provides the following information for the selected node:

- ID information on the selected node including node, subnet and domain information, as well as the application image name. Authentication keys for this node will be included only if requested. This is an option that is not recommended due to the confidential nature of the information contained within.
- If the node currently resides on hardware, the NEURON ID, the ID string, and the NEURON CHIP model number from the `read_only_data_struct` will be included. All of the communications parameter information residing in the `config_data_struct` will also be included. This includes channel ID, location string, and communication and input clocks. These structures are defined in the *NEURON CHIP Advance Information* document.

Three distinct types of address table entries may be included in the report:

- Subnet-node addressing - where the target node's subnet number and node number are used to deliver messages.
- Group addressing - where every node that belongs to a group has that group's identification in its address table.
- Turnaround addressing - where an output on the node is connected to an input on the same node. Turnaround address table entries will display their raw data values for layer 4 timing parameters and the millisecond values for those timers.

After the address table is listed, each address table entry that is dedicated to a message tag will be noted, giving the total number of tags, the number of unbindable tags, and the name of each tag.

After the address table and message tag information is displayed, the node's network variable configuration table will be displayed, ordered from the first entry to the last network variable declared.

The last entry in the report is the list of groups that are referenced by the nodes included in the report. Group information includes group name, type, size, and a full membership list of node names. The group list is included on the last page of the report.

Creating a Node Configuration Report

To create a Node Configuration Report, follow these steps:

1 From the Node Specs screen, select the nodes you want to include in your report (or Select All if all nodes are to be included). A check (✓) mark will appear in front of the selected node(s).

2 Select the Report command button. When selected, the Node Report dialog box will appear. (See Figure 11-9 below.)

3 Enter field information as follows:

Output Filename	enter a DOS filename of up to 8 characters. Letters, numbers, and underscores may be used in the name. Node Configuration Report files will always be assigned the .RPT extension.
-----------------	--

Print	if you wish to print the
Authentication	authentication keys in your
Keys	report, select the YES option. If
	you do not want these
	confidential keys to be included
	in the report, select the NO option.

4 Select buttons as follows:

OK	creates the specified report.
CANCEL	cancels the report and returns you to the Node Specs screen.

Viewing a Node Configuration Report

To view a Node Configuration Report using the LONBUILDER Editor, follow these steps:

- 1 Create the Node Configuration Report as outlined in the section titled *Creating a Node Configuration Report*, earlier.
- 2 Select F3 to access the LONBUILDER Editor.
- 3 Press *Control-O* to open a file.
- 4 Type in the full report filename including the .RPT extension, and press the Enter key.
- 5 To print the displayed report, select the Print command from the File command menu.

12

Monitoring Development Networks

This chapter describes how to monitor and analyze a development network.

The LONBUILDER Protocol Analysis Tools

The LONBUILDER Protocol Analyzer features include

- Selectively collecting, analyzing, and viewing specific network traffic based on source node, destination node, packet type, network variable, and message tag.
- Viewing network traffic statistics. You can view performance summaries, packet counts by packet type, and error rates. In addition, a network statistics status line is always visible at the bottom of the PC screen. This status line reports the
 - *Bandwidth utilization*: reflects the time used to service (media access + transmission) the transmitted data.
 - Number of packets *Received*: displays the number of packets that have been received since invoking the LONBUILDER IDE or since issuing the last *Clear* command from the Network Statistics window.
 - Number of packets *Logged*: displays the number of packets logged in the currently active packet log, or the most recently active log.

Knowledge of the message addressing schemes and the message services provided by the LONTALK protocol is helpful for making the most effective use of the protocol analyzer. Refer to the *LONTALK Protocol* engineering bulletin for an explanation of the LONTALK protocol.

Viewing Network Traffic Statistics

The protocol analyzer automatically tabulates statistics on all message traffic between network nodes that appear on the protocol analyzer channel.

To view the total network traffic statistics, follow these steps:

- 1 Press *F6* to open the Network Statistics window (see figure 12-1), or select the Statistics command from the Window menu.

≡ File Edit Search View Options Project Macro Window Help			
Network Statistics			
Start Time: Fri Aug 09 22:03:07 1991		Update Time: Fri Aug 09 22:03:48	
NETWORK PERFORMANCE			
Packets Received:	2164	Aug Packets/Sec:	52.0976
Aug Packet Size:	10.5	Max Packets/Sec:	53
Bandwidth Util:	3 %	Effective Load:	0 %
PACKET TYPES			
SESSION: Request:	0	Response:	0
TRANSPORT: Aced:	1082	Reminder:	0
		Unacked Repeat:	0
Challenge:	0	ACK:	1082
NETWORK: Unacked:	0	Reply:	0
		Unknown:	0
ERRORS			
CRC:	0	Timeout:	0
Error Rate:	0 %	Pkts Lost:	0
Update		Clear	
Bandwidth Utilization: Received: 2590 Logged: 978			

Figure 12-1. Network Statistics Window

The fields in this window are read only. They are:

Start Time

indicates the time the statistics were last cleared to 0 (zero) by starting the LONBUILDER IDE or by pressing the CLEAR button on the Network Statistics window.

Update Time

indicates the time the statistics were last updated by pressing the UPDATE button in the Network Statistics window.

Effective load is calculated as the number of protocol and data bits per second detected on the channel, divided by the bit rate of the channel. Bandwidth utilization is calculated as the number of packets per second detected on the channel, divided by the total number of available packet slots per second.

The Effective Load and Bandwidth Utilization should change correspondingly. An increase in utilization without a corresponding increase in load may indicate collisions or noise. An increase in load with no similar change in utilization may indicate that more packets are being transmitted using priority slots.

Network Performance Statistics

These fields display the number of Packets Received, the Average Packet Size, the Average Number of Packets per Second and, the Maximum number of Packets per Second since Start Time. The Effective Load and Bandwidth Utilization are instantaneous values calculated at one-second intervals. The last calculation is displayed when the screen is updated.

SESSION

These fields display statistics by packet type for protocol services implemented in the session layer of the LONTALK protocol. The fields show the number of Request, Response, and Reminder message packets propagated since Start Time.

TRANSPORT

These fields display statistics by packet type for protocol services implemented in the transport layer of the LONTALK protocol. The fields show the number of ackd, Challenge, Reply, Unackd Repeat, and ACK message packets propagated since Start Time. Various message counts which might normally be thought of as paired (ackd msg/ACK) may not show the same count. This is normal and could be due to retries, multi-cast messages, or simply the user clearing the statistics while some message transactions have not completed.

NETWORK

Error Statistics

These fields display statistics by packet type for protocol services implemented in the network layer of the LONTALK protocol. The fields show the number of Unackd and UNKNOWN messages propagated since Start Time.

These fields display the number of CRC errors, time out errors, and packets lost, as well as the error rate since start time.

Packets lost reflect those packets that the protocol analyzer missed (did not buffer due to overflow of internal buffers).

These errors are determined by the protocol analyzer and may not necessarily reflect the errors detected by other nodes on the channel.

A CRC error is a packet that fails the cyclic redundancy check, possibly due to a collision.

A lost packet is a packet for which no buffer was available.

A timeout occurs when the bit sync is detected, but no packet follows.

This is usually due to a noisy channel, or incorrect bit rate.

2 Select buttons as follows:

Update

updates the statistics to the current time.

Clear

sets the statistics to 0 (zero).

Collecting and Viewing Message Packets

The LONBUILDER protocol analyzer allows you to:

- Create packet logs for collecting packets. The protocol analyzer places the packet logs in a log directory that is subordinate to the project directory. You use the protocol analyzer to view all the packets in a log, as well as examine each packet individually. You cannot print a packet log.
- Define the packet log filter to filter packets collected in a log by packet type, source node, destination node, network variable, and message tag.

Creating Packet Logs

To create a packet log, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network Mgmt button to list the Network Management object type selectors—Target HW, NV Browser, Packet Log, and Statistics.
- 3 Select the Packet Log button to display the existing packet logs, their on/off status, and the number of packets in them. All logs might be off; only one log can be on.
- 4 Select the Create command button. A dialog box opens prompting for the name of the log.
- 5 Enter the log name. (The protocol analyzer appends the .LOG extension to the name you enter.)
- 6 Select buttons as follows:

OK	saves the log name and returns to the Navigator window, where the log name is displayed with its status off.
----	--

CANCEL	cancels the prompt and returns you to the Navigator window.
--------	---

Turning On a Packet Log

The protocol analyzer logs packets to only one log at a time. When no log is turned on, no packets are logged.

To turn packet logs on and off, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network Mgmt button to list the Network Management object type selectors—Target HW, NV Browser, Packet Log, and Statistics.
- 3 Select the Packet Log button to display the existing packet logs, their on/of status, and the number of packets in them. All logs might be off; only one log can be on.
- 4 Select a packet log from the displayed list.
- 5 Select the ON/OFF command button to turn on the selected log, if it is off, or to turn off the selected log, if it is on.

You can also turn the current packet log on and off from the Protocol Analyzer window. See Viewing a Packet Log, later, for details.

If the protocol analyzer detects a disk full condition while you are logging packets, logging will no longer take place and packets will be dropped from the logging buffer. There will be no error message displayed. You may detect this condition if you have the log turned on and there are packets being received, but the logged counter is not incrementing. Note that similar symptoms may occur if packet filtering is on, depending on the filter and the nature of the packets received. Use normal DOS commands to check if your disk is full.

The packet log filter allows you to select the packets you want to filter. The packets are collected in a log according to specific packet types, source nodes, destination nodes, network variables, and message tags.

You can define the packet log filter while using the Protocol Analyzer window and the Statistics window. You can turn the filter on and off as needed.

To define the packet log filter, follow these steps:

- 1 From the Protocol Analyzer or the Network Statistics window, open the Options pull-down menu.
- 2 Select the Filters option to open the Log Filter dialog box (see figure 12-2). This dialog box has four subwindows: Packet Types, Source Nodes, Destination Nodes, and Variables and Tags. In addition, there are seven command buttons: Add Node, Add NV, Add Tag, Delete, Clear, OK, and Cancel, and two toggle switches: Filter ON/OFF, and AND/OR.

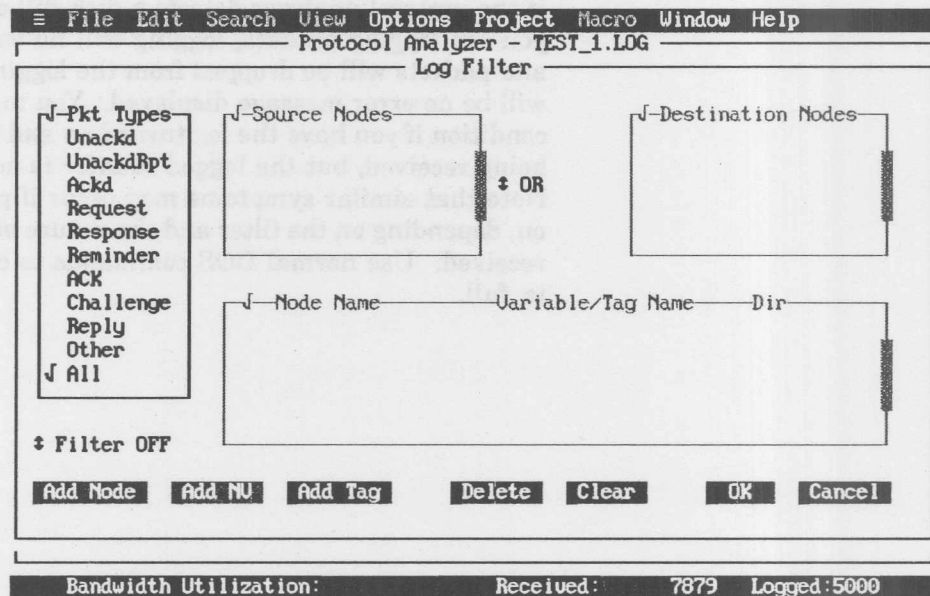


Figure 12-2. Log Filter Dialog Box

3 Add nodes, network variables and message tags to the packet log filter by selecting the command buttons as follows:

Add Node	to add nodes to the packet log filter.
Add NV	to add network variables to the packet log filter. (Only connected network variables are available.)
Add Tag	to add message tags to the packet log filter. (Only connected message tags are available.)
Delete	to delete a node, network variable or tag, select the item. A check mark (✓) will appear in front of the selected item. Press the Delete button.

4 Information will be displayed in the subwindows as follows:

The maximum number of nodes which can be added to a node window is 16.

Packet Types	lists the packet types supported by the LONTALK protocol. (Only the selected packet types will be collected when the log is turned on.)
Source Nodes	lists the nodes, added as source nodes to the filter by selecting the Add Node command. If a node is added to the source node window, messages from this node, which satisfy all other filter conditions, will be collected.
Destination Nodes	lists the nodes added as destination nodes to the filter by selecting the Add Node command. If a node is added to the destination node window, all incoming messages to this node, which satisfy all other filter conditions, will be collected.

The maximum number of tags or network variables which can be added to the variable window is 32.

Variables/Tags

lists the network variables and message tags added to the filter by selecting the Add NV or Add Tag commands.

If a node (source/destination) is selected and a network variable (output/input) on that node is also selected, the network variable is a redundant element. The filter will collect for all network variables on that node.

To select an input NV to add to the filter, follow these steps:

- Select the Add NV button from the main filter window.
- Choose the input direction.
- Choose a node from the scrollable list.
- Choose a network variable from the Net Var list (only those connected network variables of the selected direction are displayed.)
- Select the Add button.

To add output network variables to the filter, choose the output direction in step 2 above, and repeat the last three steps listed above.

For input network variables added to the filter, all updates and polls to that network variable, and responses to the polls, will be collected. For output network variables, all updates and polls from that node's network variable will be collected.

To select a message tag to add to the filter, follow these steps:

- Select the Add NV button from the main filter window.
- Choose the desired direction. Unlike network variables which have a direction, message tags (except Msg-In tags) are directionless. In choosing a direction, you are determining which messages will be filtered by the protocol analyzer. When the direction is Msg-Out, all messages from the specified node, for that tag, will be collected. If the direction is Msg-In, messages from all senders to the specified tag will be collected.

For tag connections less than 3, the protocol analyzer is unable to uniquely identify the message. It will collect for all messages (not network variables) using subnet node addressing.

- Choose a node from the scrollable list.
- Choose a tag from the scrollable list.
- Select the Add button.

5 Select buttons as needed:

Filter Operation toggles the FILTER OPERATION field between AND and OR.

AND means the filter collects a message packet only if the message satisfies two conditions: it must be sent from one of the selected source nodes **and** it must be sent to at least one of the selected destination nodes. (In both **and/or** cases, the packet type must satisfy the packet type filter.)

	OR	means the filter collects a message packet if the message satisfies one of two conditions: it must be sent from one of the selected source nodes or it must be sent to at least one of the selected destination nodes. (In both and/or cases, the packet type must satisfy the packet type filter.)
Filter ON/OFF		toggles the Filter field between ON and OFF.
CLEAR		clears all selections from the packet log filter window so you can start over.
OK		saves and commits the created packet log filter.
CANCEL		returns you to the previous window without saving any changes to the existing packet log filter.

Viewing a Packet Log

You can view the packets in a packet log that is on and collecting packets or that is off and has collected packets in the past. To view the packets in a packet log, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network Mgmt button to list the Network Management object type selectors—Target HW, NV Browser, Packet Log, and Node Specs.
- 3 Select the Packet Log button to display a list of existing packet logs, their on/off status, and the number of packets in them. All logs might be off; only one log can be on.

If you open the Protocol Analyzer window using the accelerator key (F5), or the Window pull down menu key, and you have not previously selected a packet log, the Protocol Analyzer will display the default UNTITLED.LOG, and will create this log if it does not currently exist.

- 4 Select a log from the displayed list. The selected log can be on or off.
- 5 Select the List button to open the Protocol Analyzer window (see figure 12-3) with the logged packets, if any, listed in the top window, and with the first logged packet detailed in the bottom window. The name of the selected log is displayed in the border title of the window. The name of the current log file is displayed in the LOGFILE field at the bottom of the window. The current log file is the log that is either currently on or that has been on most recently.

From the Protocol Analyzer window, you can:

- Move to the beginning and end of the packet list with the *Home* and *End* keys.
- Move a page at a time with the *Page Up* and *Page Down* keys.
- Quickly locate a packet log by selecting the *Go To* option from the *Window* command menu, and entering the packet number you want to locate. If you do not know the desired packet number, use the + and - options to move forward and backward through the list of available packets.
 - (number) enter a minus sign (-) followed by a number to back-up through the list of available packets.
 - + (number) enter a plus sign (+) followed by a number to move forward through the list of available packets. Use the *Home* and *End* keys to quickly move to the beginning or the end of the log.
- Turn the current log file (the file displayed in the LOGFILE field at the bottom of the window) on or off by selecting the LOG ON/OFF button.

- Turn the existing filter on or off by selecting the FILTER ON/OFF button. (The filter only applies to those packets logged to the active file.)

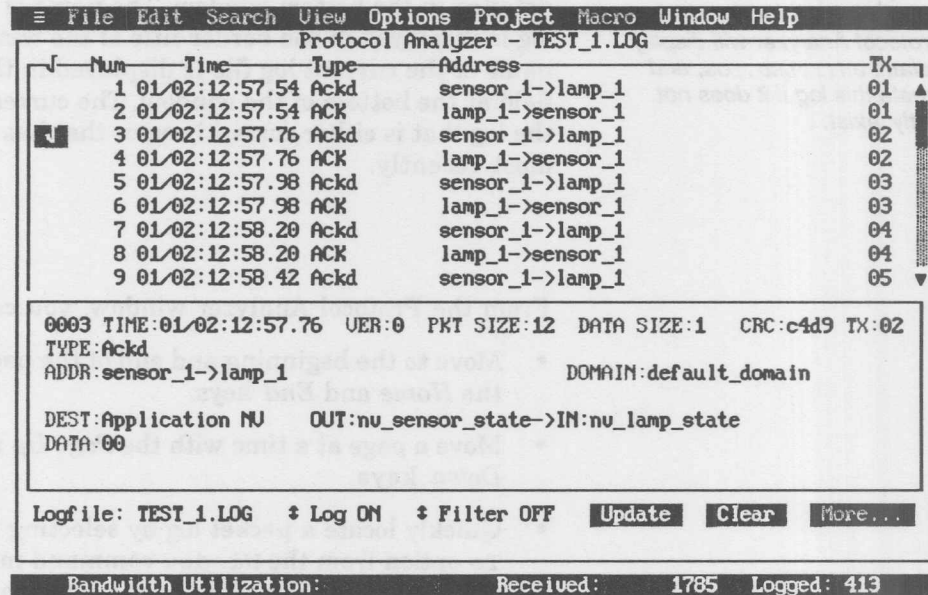


Figure 12-3. Sample Protocol Analyzer Window

In figure 12-3, the selected packet was sent from sensor_1.nv_sensor_state (where sensor_1 is the node and nv_sensor_state is the network variable) to lamp_1.nv_lamp_state. To examine a different packet in detail, select it. The bottom of the Protocol Analyzer window displays the detailed information about the selected packet.

In the sample Protocol Analyzer window in figure 12-4, the selected packet was sent from sensor_2.nv_sensor_state to lamp_2.nv_lamp_state, lamp_3.nv_lamp_state, and lamp_4.nv_lamp_state. The binder automatically places nodes in groups when there are multiple input network variables or message tags in a connection.

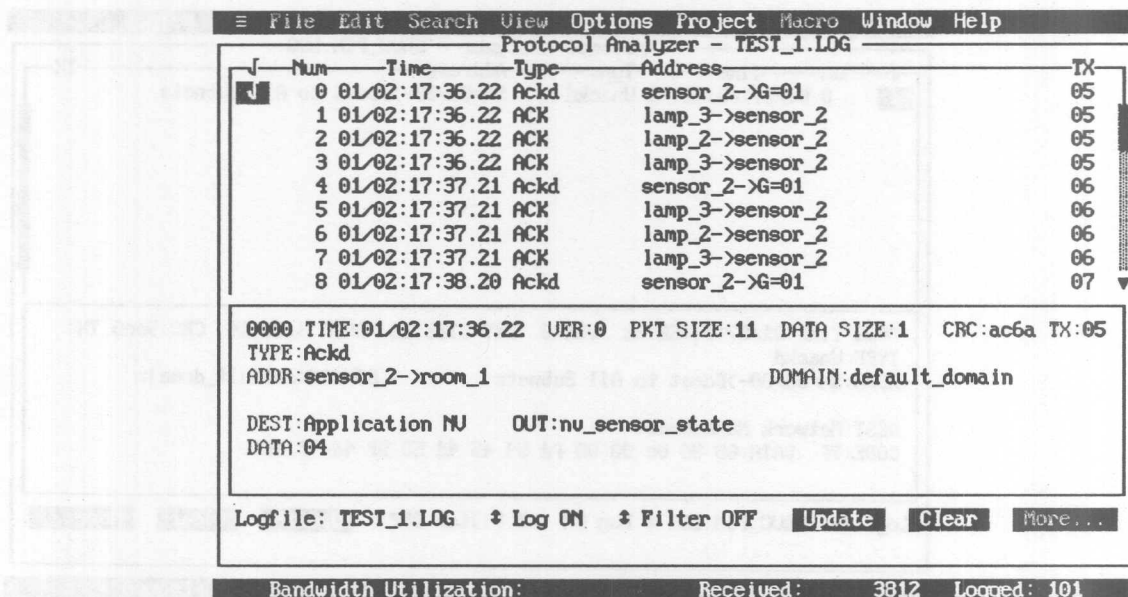


Figure 12-4. Sample Protocol Analyzer Window

The sample Protocol Analyzer window in figure 12-5 shows a network management message. The message shown is a service pin message broadcast to all nodes in response to a user pressing the service pin on a node.

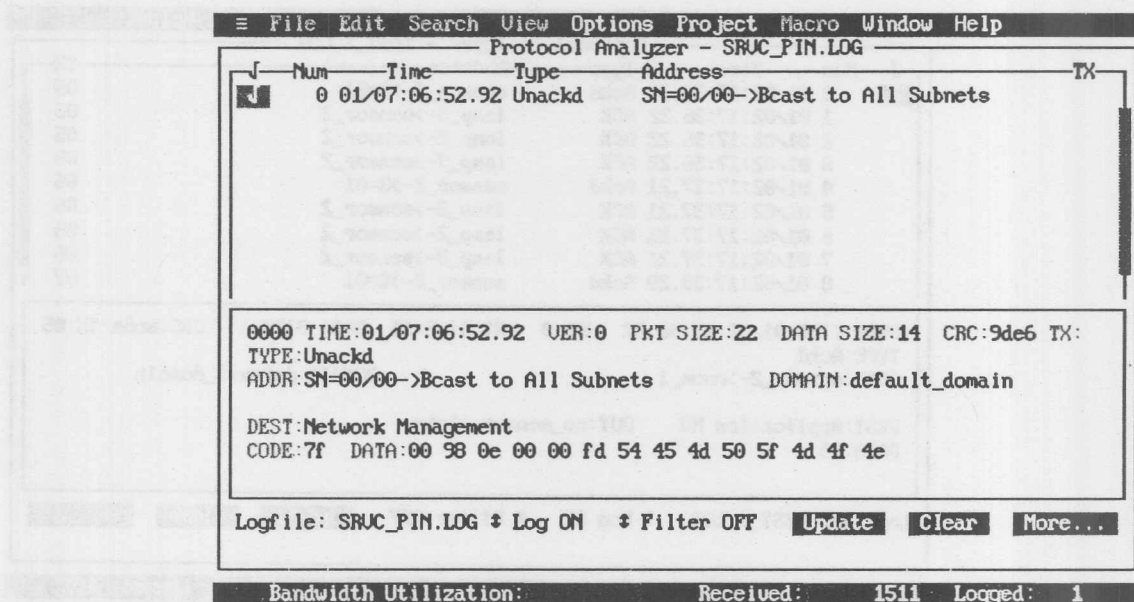


Figure 12-5. Sample Protocol Analyzer Window

6 Select buttons as needed:

LOG

toggles the LOG field between ON and OFF.

ON means the current log file is collecting data.

OFF means the current log file is not collecting data.

FILTER

toggles the FILTER field between ON and OFF.

ON means the filter is ON.

OFF means the filter is OFF.

UPDATE

when the current log file is displayed, this field updates the screen to display the latest logged packets (whether the log file is ON or OFF). The Log indicator in the status line displays the current number of logged packets.

CLEAR LOG

clears the current log file.

MORE

provides an expanded display of the selected packet. This field is used for packets that cannot be displayed completely in the Protocol Analyzer window, for network management commands, and for packets with CRC errors.

F5 is the accelerator key that opens the Protocol Analyzer window. To open this window at any time and view the current packet log, press F5. The Protocol Analyzer window opens (see figure 12-3, earlier) with the contents of the current packet log displayed. The current packet log is the log that is either currently on or that has been on most recently. If you open the Protocol Analyzer window using the accelerator key (F5), or the Window pull down menu key, and you have not previously selected a packet log, the protocol analyzer will display the default UNTITLED.LOG, and will create this log if it does not currently exist.

Network Management Message Interpretation

The protocol analyzer can interpret network management and diagnostic commands and responses.

All packet data will be displayed in "raw" form in the full packet display in the Protocol Analyzer window. However, the More command will be enabled in the following cases:

- 1 The protocol analyzer identifies the packet as a network management command.

Refer to Appendix B in the NEURON CHIP Advance Information document, for information on data structures associated with network management and diagnostic commands.

- 2 The protocol analyzer identifies the packet as a network diagnostic command.
- 3 The protocol analyzer identifies the packet as a response. After activating the More button, you will be given three options for viewing this packet: as a response to an Application Message, a Network Management Command, or a Diagnostic Command. You can decide how to view the message by selecting one of the three command buttons.

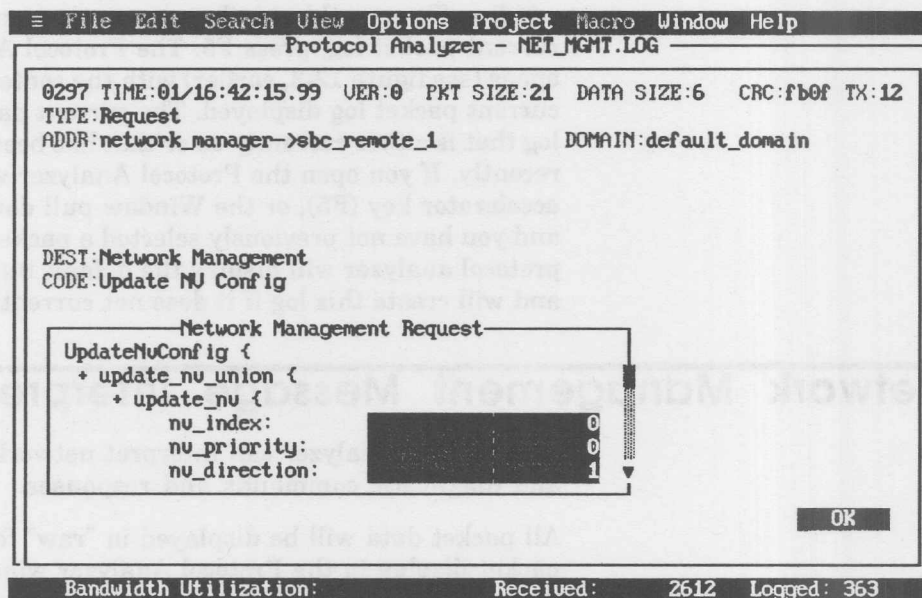


Figure 12-6. The More Window for Requests

Refer to the `netmgmt.h` file, in your LONBUILDER include directory, for similar type definitions for most network management commands.

The network management message will be displayed in a NEURON C-like structure format.

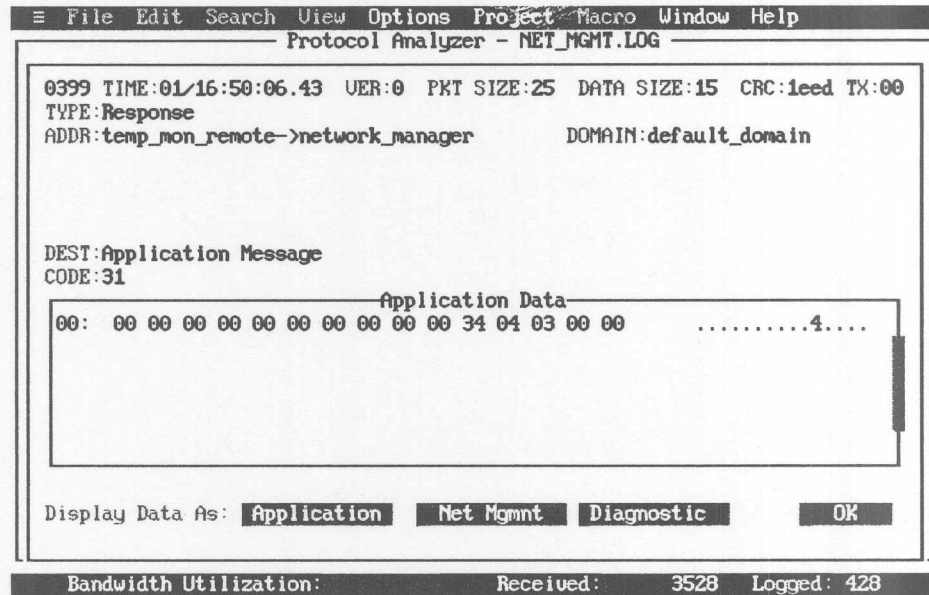


Figure 12-7. The More Window for Responses

13

Testing Development Networks

This chapter describes how the LONBUILDER Network Manager is used to test routers and application nodes on a development network.

LONBUILDER Network Manager Test Commands

The LONBUILDER Network Manager includes commands to

- Test target hardware over the network
- Take application nodes and routers on and off line
- Reset application nodes and routers
- Wink nodes
- Browse node memory
- Browse network variables

All of these commands are issued from the Target Hardware window. Figure 13-1 shows a sample Target Hardware window with emulator, SBC, and custom node target hardware objects displayed in the object list.

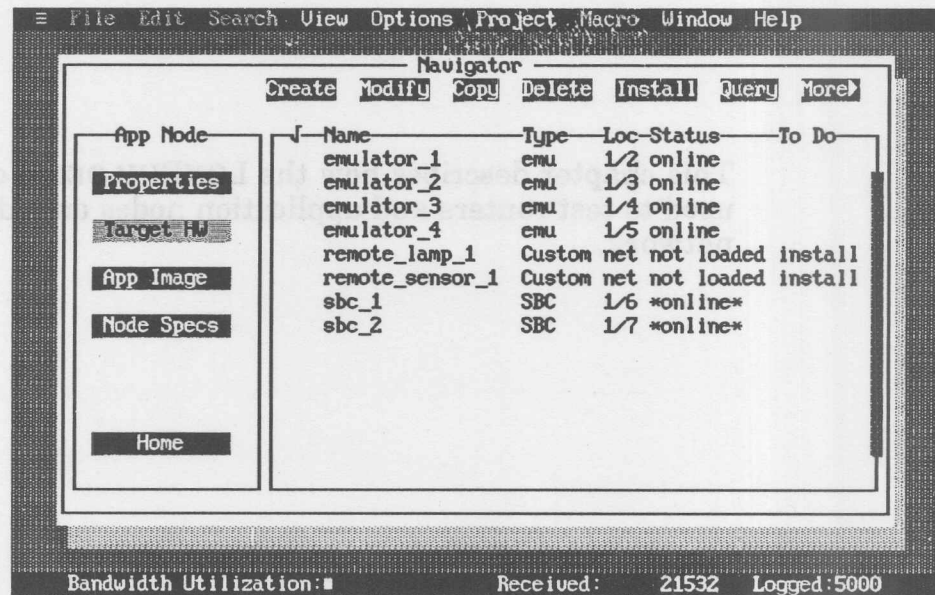


Figure 13-1. Target Hardware Window

In addition to its name, the object list also displays the location (Loc) of the target hardware, its hardware status (Status), and its to-do state (To Do).

Loc is either the string net or a development station ID/slot number.

net

indicates that the node is located remote from a development station.

development station ID/slot number

indicates the development station and slot within it where the node is located.

Status is the hardware status of the node. The status of an emulator, with an application whose origin is source code, can be:

flush

node is currently flushing messages prior to sleep mode. Refer to the *NEURON C Programmer's Guide* for more information on this mode.

halted

node is halted.

no response

node is not responding to network management test commands.

not loaded

node has not been loaded with its application image.

app'less

The node has been loaded, but is currently in the applicationless state. This may be caused by a network management tool that has put the node into the applicationless state. It will also occur if application EEPROM is corrupted by the application. If corruption is suspected, look in the debugger error log to see if an application checksum error has been logged.

unconfig'd

The node has been loaded, but is currently in the unconfigured state. This may be caused by a network management tool that has put the node into the unconfigured state. It will also occur if configuration EEPROM is corrupted by the application. If corruption is suspected, look in the debugger error log to see if a configuration checksum error has been logged.

offline

node is offline.

online

node is online.

preempt

node is in pre-emption mode. Refer to the *NEURON C Programmer's Guide* for more information on this mode.

protect

node is halted due to a read or write protect violation.

sleep

node is asleep.

The status of an SBC, a custom node, and an emulator with an imported application image can be:

not loaded

node has not been loaded with its application image.

offline

node was offline as of the last operation on the node. Because of its remote location on the network, the offline status may not be current.

online

node was online as of the last operation on the node. Because of its remote location on the network, the online status may not be current.

app'less

the node has been loaded, but is currently in the applicationless state. This may be caused by a network management tool that has put the node into the applicationless state. It can also occur if the application EEPROM is corrupted by the application.

unconfig'd

the node has been loaded, but is currently in the unconfigured state. This may be caused by a network management tool that has put the node into the unconfigured state. It can also occur if the application EEPROM is corrupted by the application.

no response

node does not respond to network management test commands.

If an SBC or custom node has a node state of *applicationless* or *unconfigured*, the LONBUILDER system may not be aware of this and may display *online* or *offline* in the Target Hardware status display. If a Test command is issued to the node, the status will then indicate the actual state. In either case the To Do field will not necessarily indicate that the node must be loaded.

Testing Application Nodes

The test command reads and displays the error table statistics of target hardware. To test target hardware, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the **App Node** button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the **Target HW** button to display the existing target hardware objects in the object list.
- 4 Select one or more of the displayed nodes.
- 5 Select the **Test** command button to send the *query status* network management message and report the results in the Hardware Test Results window (see figure 13-2).

Selecting the More button displays additional command buttons in the subwindow.

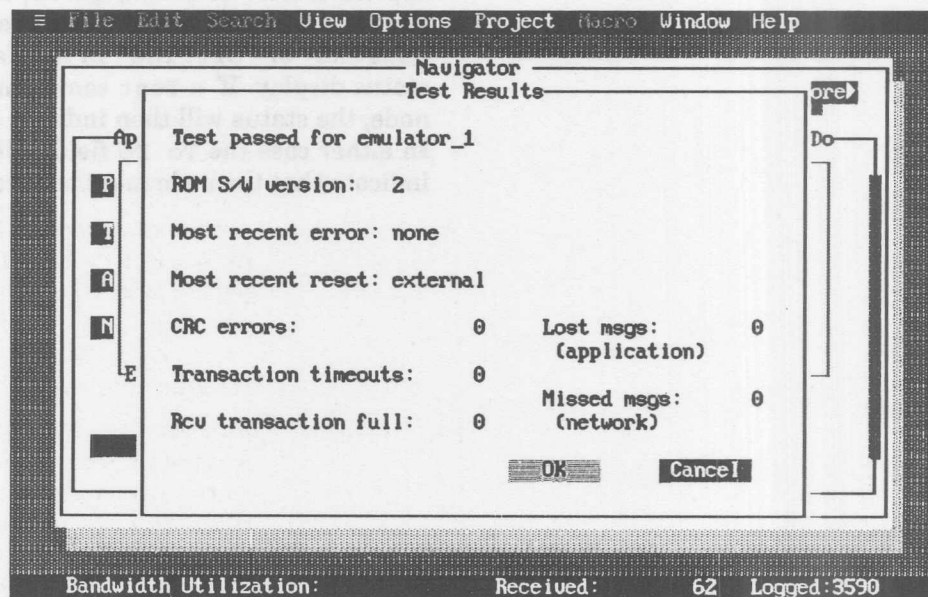


Figure 13-2. Hardware Test Results Window

The fields in the Hardware Test Results window are read-only. They are:

ROM S/W version	version number of the node's NEURON CHIP firmware.
Most recent error	most recent system or application error logged by the NEURON CHIP.
Most recent reset	states how the node was last reset. The reset can be:
External	last reset by an external source such as the reset button.
Software	last reset by executing a reset clause.
External or Software	last reset by an external source such as the reset button, or executing a reset clause.
Power Up	last reset by being powered up.
Watch-dog Timer	last reset by the watch dog timer timing out.
CRC errors	number of CRC errors that have occurred on the network, detected during packet reception. This could result from a collision, a noisy medium, signal attenuation, etc.
Lost Msgs	number of messages addressed to the node that were thrown away because there was no application buffer available for the messages. You can set the number of application buffers at compile time.

Transmission
timeouts

number of timeouts that have occurred in attempting to carry out acknowledged or request/response transactions. A timeout occurs when a node fails to receive all the expected acknowledgements or responses after retrying the configured number of times at the configured interval.

Missed Msgs

number of messages that were on the network but could not be received because there was no network buffer available for the message. You can set the number of network buffers at compile time.

Rcv transaction
full

number of times an incoming unack_rpt, ackd, or request message was lost because there was no more room in the receive transaction database. You can set the size of this database at compile time, with the NEURON C compiler directive `receive_trans_count`.

Testing Routers

The test command can also be used to read and display the error table statistics of routers. To test routers, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Router button to list the router object type selectors—Target HW and Node Specs.
- 3 Select the Target HW button to display the existing router target hardware objects in the object list.
- 4 Select one or more of the displayed routers.
- 5 Select the Test command button.
- 6 The Hardware Test Results window will appear once for each router side. The fields in these windows are read-only. Refer to the field descriptions in this chapter under *Testing Application Nodes*.

Taking Application Nodes On and Off Line

The Online and Offline commands send a *set node mode* network management message to put the node online or offline and to execute an online or offline clause, if there is one, in the node's application program. Refer to the *NEURON C Programmer's Guide* for details on programming an online and offline clause.

To take application nodes online and offline, follow these steps:

Selecting the More button displays additional command buttons in the subwindow.

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the Target HW button to display the existing target hardware objects in the object list.
- 4 Select one or more of the displayed nodes.
- 5 Select the Online or Offline command button.

The node will execute its when (online) clause whenever it transitions from the offline state to the online state. Similarly, it will execute its when (offline) clause when it transitions to the offline state.

Taking Routers On and Off Line

To take routers online and offline, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Router button to list the router object type selectors—Target HW and Node Specs.
- 3 Select the Target HW button to display the existing router target hardware objects in the object list.
- 4 Select one or more of the displayed routers.
- 5 Select the Online or Offline command button.

Selecting the More button displays additional command buttons in the subwindow.

Resetting Nodes

The reset command sends a *set node mode* network management message to reset the target hardware and to execute its application. To reset target hardware, either

Press the reset button on the node.

or

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the Target HW button to display the existing target hardware objects in the object list.
- 4 Select one of the displayed nodes.
- 5 Select the Reset(not the Reset/Halt) command button.

Selecting the More button displays additional command buttons in the subwindow.

Resetting Routers

The reset command sends a set node mode network management message to reset the router and to execute its firmware. To reset routers, either

Press the reset button on the router.

or

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Router button to list the router object type selectors—Target HW and Node Specs.
- 3 Select the Target HW button to display the existing router target hardware objects in the object list.
- 4 Select one or more of the displayed routers.
- 5 Select the Reset command button.

Selecting the *More* button displays additional command buttons in the subwindow.

Winking Nodes

The wink command sends a *wink* network management message to activate a wink clause, if there is one, in the node's application program. Refer to the *NEURON C Programmer's Guide* for details on programming a wink clause.

To send a wink command, follow these steps:

- 1 Press *F2* to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the Target HW button to display the existing target hardware objects in the object list.

Selecting the **More** button displays additional command buttons in the subwindow.

- 4 Select one of the displayed nodes.
- 5 Select the **Wink** command button.

The acknowledgement of the wink message is sent back to the network manager when the message is delivered to the application program, and not when the execution of the wink clause is completed.

Browsing Node Memory

The network manager allows you to read and write, over the network, the memory in any application node (unless the node has been loaded with an application that is read and write protected). The memory of processor boards installed inside development stations may optionally be read over the development station backplane. Using the detailed link map for the node generated during a build (refer to the section on *Configuring LONBUILDER Project Parameters* in Chapter 4 and Appendix C.), you can browse node memory and modify node parameters.

You can also use the memory browser to read from and write to memory mapped I/O devices, provided this area is configured in the node's hardware properties. Memory-mapped accesses must always be done over the network.

The following is a sample from the **SYMBOLS BY NAME** section of a detailed link map. It shows the memory locations and the network variable names preceded by a percent sign (%):

```
. EFFF %flag_a
. EFFE %flag_b
. EFFD %flag_c
. EFF8 %offset_a
. EFF6 %offset_b
. EFF4 %offset_c
. EFFC %count_a
. EFFB %count_b
. EFFA %count_c
. EFF2 %ula
. EFF0 %ulb
. EFEE %ulc
```

To browse node memory, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the App Node button to list the application node object type selectors—Properties, Target HW, App Image, and Node Specs.
- 3 Select the Target HW button to display the existing target hardware objects in the object list.
- 4 Select one or more of the displayed nodes.
- 5 Select the Memory command button. The memory browser for the first target hardware object selected displays in the Target Hardware Memory window (see figure 13-3).

Selecting the More button displays additional command buttons in the subwindow.

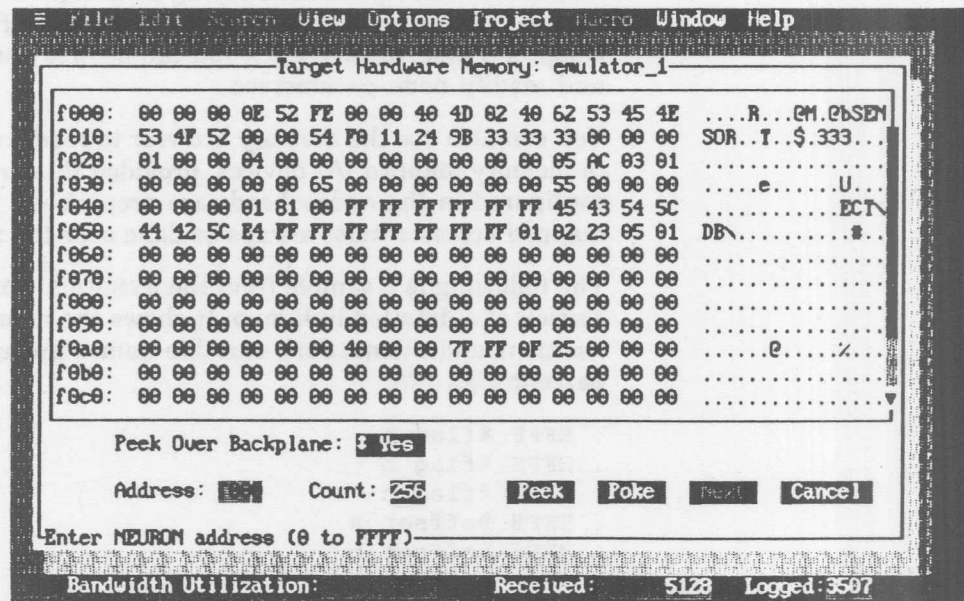


Figure 13-3. Target Hardware Memory Window

- 6 Select the ADDRESS field and enter a memory address.
- 7 Select the COUNT field and enter a byte count, up to 256.
- 8 If the target hardware is installed in a development station (its location is not 0/0), select the Peek Over Backplane field to control whether reads are done over the backplane or over the network. Writes are always done over the network.
- 9 Select the PEEK button to read memory over the backplane (if peek over backplane is yes), or to send a *read memory* network management message (if peek over backplane is no) and display the results.
- 10 Select the displayed contents and change it as desired.
- 11 Select the POKE button to send a *write memory* network management message to write the changed contents to the memory address. (Writing memory is always done over the network.)
- 12 Select buttons as follows:

NEXT	displays the memory browser for the next node selected.
CANCEL	returns you to the Navigator window without further modifying the node's memory.

Multi-Channel Network Management Commands

Network management commands are sent over the network using the network manager node. The network manager may require the services of one or more routers to communicate with a given target node. All of the routers in the communication path between the network manager and target node must have a domain in common, unless a router installed as a repeater is being used.

If the target node is configured, it must also be in a domain common to all of the routers. The LONBUILDER network manager confirms that the routers and target nodes share a domain. If this is not the case, an error message is displayed prior to attempting to send the message. However, the LONBUILDER network manager does not verify that all of the appropriate routers are currently loaded and online.

The Network Variable Browser

The Network Variable (NV) browser is a network management tool which provides online access to network variable information for any application node in a development network. The NV browser function enables you to create lists of selected network variables and monitor, modify, or update these variables during operation:

VIEW	displays both input and output network variables from the browser by polling specified network variables.
MODIFY	changes specified input network variables by sending updates to the input network variables from the browser.
UPDATE DATA	polls selected nodes and displays the first field of the selected network variable.

Using the Network Variable (NV) Browser

From the Navigator NV browser object display you can Create or Delete a browser list, or select the list or lists you wish to browse:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network Mgmt button to list the network management object type selectors—Target HW, NV Browser, Packet Log, and Statistics.
- 3 Select the NV Browser button to display existing browser lists.

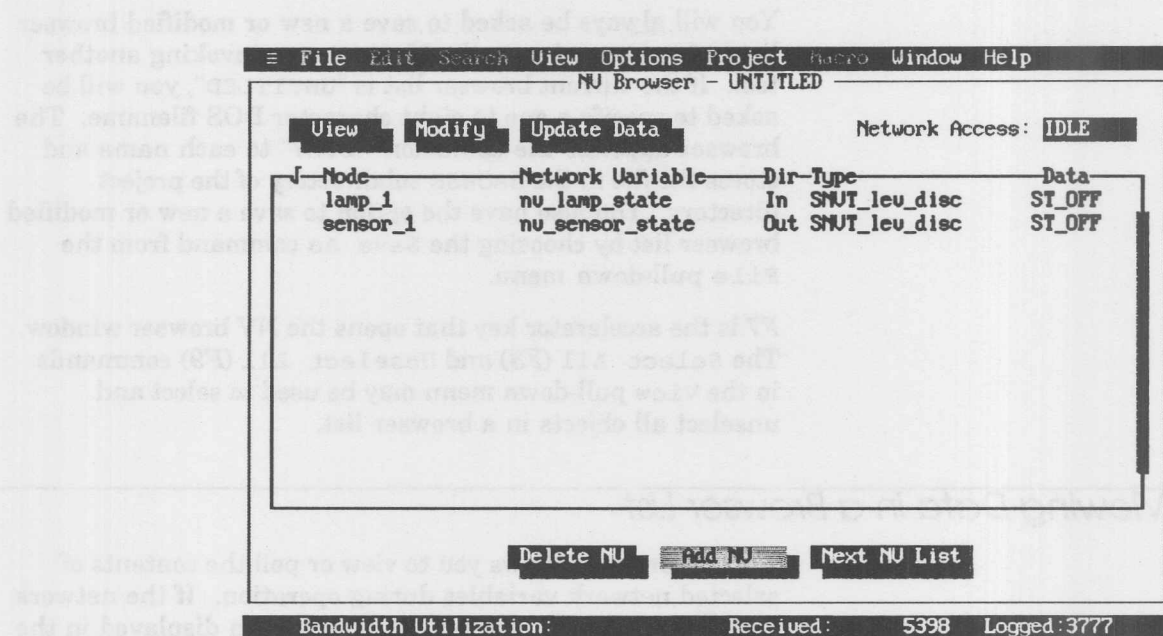


Figure 13-4. Network Variable (NV) Browser

4 Select the command buttons as follows:

CREATE

to create a new browser list . The NV browser window opens with an untitled browser list. Select the ADD NV command button to add network variables to the list (see figure 13-4).

BROWSE

to browse the network variables in the browser list. Select one or more lists you want to browse. A check mark (✓) appears to the left of the selected list.

DELETE

to delete the specified browser list. Select one or more lists you want to delete. A check mark (✓) appears to the left of the selected list.

You will always be asked to save a new or modified browser list before returning to the Navigator or invoking another tool. If the current browser list is "UNTITLED", you will be asked to specify a one to eight character DOS filename. The browser appends the extension ".BRW" to each name and stores the file in the BROWSE subdirectory of the project directory. You also have the option to save a new or modified browser list by choosing the Save As command from the File pull-down menu.

F7 is the accelerator key that opens the NV browser window. The Select All (**F8**) and Unselect All (**F9**) commands in the View pull-down menu may be used to select and unselect all objects in a browser list.

Viewing Data in a Browser List

The NV browser allows you to view or poll the contents of selected network variables during operation. If the network variable is a SNVT, the type of information displayed in the browser list will be obtained from the SNVT definition. If the network variable is not a SNVT type, then the contents will be displayed in the format provided by the abbreviated type information for each network variable in the database.

To view the data in a browser list, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network Mgmt button to list the network management object type selectors—Target HW, NV Browser, Packet Log, and Node Specs.
- 3 Select the NV Browser button to display existing browser lists, and select the list or lists you want to view. A check mark (✓) appears to the left of the selected list.
- 4 Select the BROWSE command button to open the NV browser window.

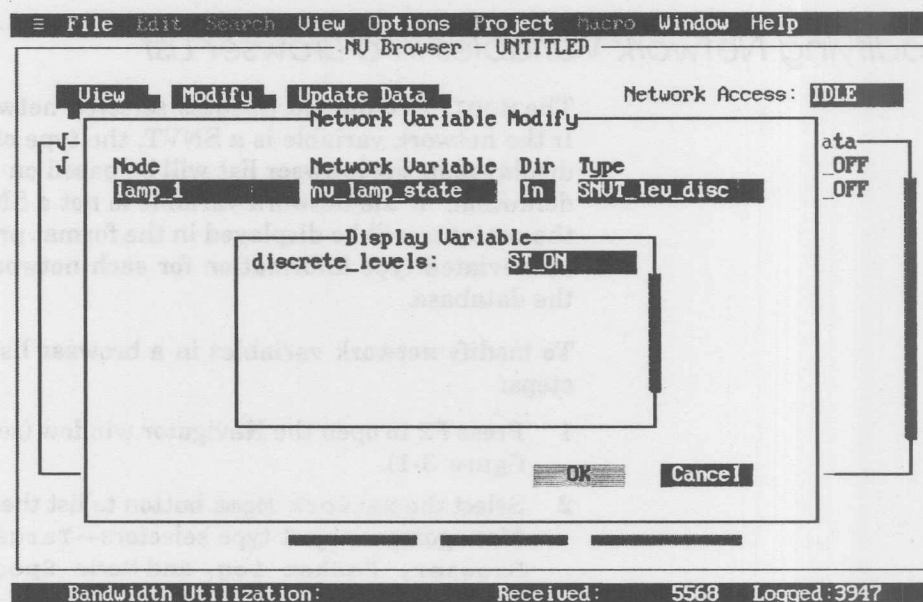


Figure 13-5. NV Browser Window and Display Variable dialog box

The NV browser obtains the value of a network variable by polling the remote node, and modifies the value by sending an update. The "Network Access" field in the main window provides feedback to the user when polling or updating is in progress.

- 5 Select the variable you want to view. A check mark (✓) appears to the left of the selected variable.
- 6 Select the VIEW command button to open the Network Variables display and view node, variable, direction, and type information.
- 7 To view additional information regarding this variable, click on the data fields in the Display Variable window. This opens the Modify Variable dialog box which contains type information and alternate display types. (VIEW only allows you to look at this information, not change it. To change this information, refer to the next section on Modifying Network Variables in a browser list.)
- 8 Select the CANCEL button to close the dialog box.

Modifying Network Variables in a Browser List

The MODIFY command updates selected network variables. If the network variable is a SNVT, the type of information displayed in the browser list will be based on the SNVT definition. If the network variable is not a SNVT type, then the contents will be displayed in the format provided by the abbreviated type information for each network variable in the database.

To modify network variables in a browser list, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the **Network Mgmt** button to list the Network Management object type selectors—Target HW, NV Browser, Packet Log, and Node Specs.
- 3 Select the **NV Browser** button to display existing browser lists and select the list you want to modify. A check mark (✓) appears to the left of the selected list.
- 4 Select the **BROWSE** command button to open the NV browser window for the first selected list.
- 5 Select the variable you want to modify. A check mark (✓) appears to the left of the selected variable.
- 6 Select the **MODIFY** command button to open the Network Variable Modify window and access node, variable, direction, and type information (see figure 13-6).

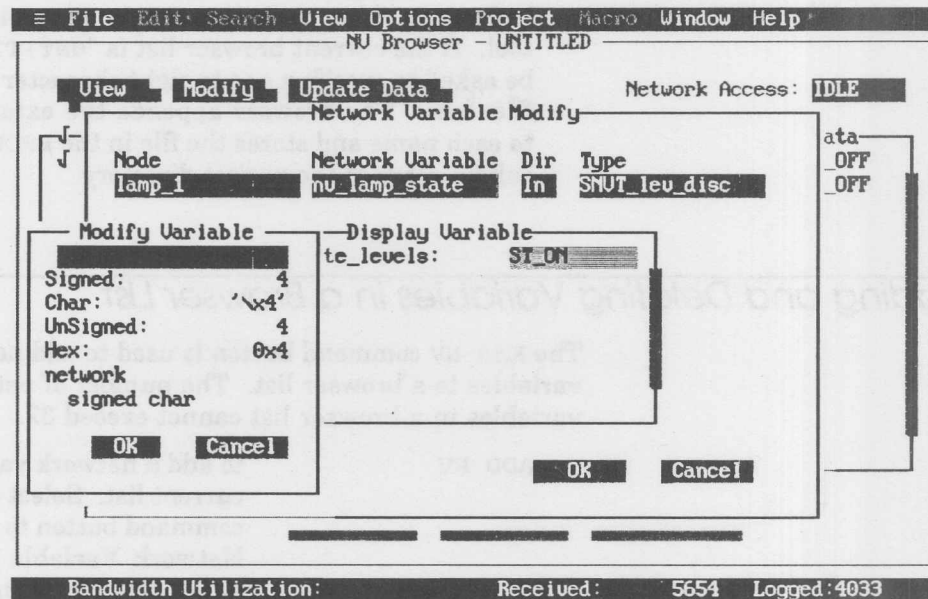


Figure 13-6. NV Browser Window and Modify Variable dialog box

- 7 To change the specified network variable, select the data value or type into this field to open the Modify Variable dialog box (see figure 13-6) The dialog box contains type information and alternate display types, for example: char, hex. (A SNVT enum type variable is displayed with the appropriate enum identifier, if one exists.)
- 8 Enter the new value for the variable. Data entry follows NEURON C syntax.
- 9 Select the OK button to accept the new value and exit the dialog box. The network variable will not be changed until the OK button is selected. SNVT data fields will be converted to appropriate units at this time also. If the data is out of the range of the SNVT type, then the data field in the Display Variable dialog box will be indicated in red. To cancel this operation and exit the dialog box, select the CANCEL button.

- 10 You will always be asked to save a modified browser list before returning to the Navigator or invoking another tool. If the current browser list is "UNTITLED", you will be asked to specify a one to eight character DOS filename. The browser appends the extension ".BRW" to each name and stores the file in the BROWSE subdirectory of the project directory.

Adding and Deleting Variables in a Browser List

The Add NV command button is used to add network variables to a browser list. The number of network variables in a browser list cannot exceed 32.

ADD NV

to add a network variable to the current list. Select the ADD NV command button to open the Add Network Variable window. Select fields and enter new values as needed. Select the START button to accept these selections, or the ADD NEXT button to accept these selections and add the next variable.

DELETE NV

to delete a network variable from the current list. Select one or more network variables from the current list. Check marks (✓) appear to indicate your selections. Select the DELETE NV command button.

Updating Data in a Browser List

The NV browser obtains the value of a network variable by polling the remote node via subnet/node addressing. The displayed data may appear to be inconsistent due to the serial polling of each node. For example, suppose a list contains one output and three input network variables, and they are all part of one connection. When you select all four for updating, they may contain different values depending on the exact time they responded to the poll.

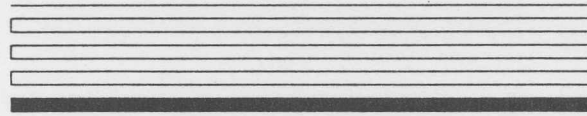
The UPDATE DATA command allows you to update selected input network variables.

To update data in a browser list, follow these steps:

- 1 Press **F2** to open the Navigator window (see Chapter 3, figure 3-1).
- 2 Select the Network Mgmt button to list the Network Management object type selectors—Target HW, NV Browser, Packet Log, and Node Specs.
- 3 Select the NV Browser button to display existing browser lists and select the list to update. A check mark (✓) indicates the selection.
- 4 Select the BROWSE command button to open the NV browser window for the first selected list.
- 5 Select the variable or set of variables you want to update. A check mark (✓) appears to the left of the selected variable.
- 6 Select the UPDATE DATA command button to update the selected variables. The Network Access indicator (IDLE), in the upper right corner of the NV Browser window changes to indicate that the network is being polled (POLLING) and that data is being updated (UPDATING).

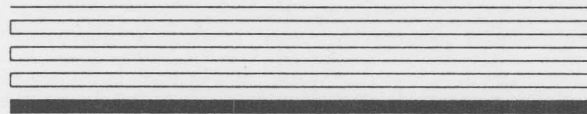
The first field of the Network Variable will be updated in the data field on the main NV browser list. If the NV browser was unsuccessful in polling or updating the data from the specified node, the shorter data field will contain "????????".

Network management authentication must be enabled to use the Network Variable Browser to update network variables with authentication enabled. Network management authentication is enabled on the Node Create or Node Modify dialog box. Network variable authentication is enabled on the Connection Parameters dialog box.



Part 4

Appendices



A

Main Menu Bar

This appendix defines the commands in the pull-down menus on the LONBUILDER main menu bar. How to use the cursor or arrow keys to open and close a pull-down menu and to select a command from an open menu is explained in Chapter 3, *Introduction to the User Interface*.

Menu Commands

There are ten pull-down menus on the main menu bar: System, File, Edit, Search, View, Options, Project, Macro, Window, and Help.



As an alternative to opening and selecting a command from pull-down menus with the mouse or with arrow keys, you can use an accelerator key. This section lists the accelerator keys that open the pull-down menus and invoke the commands on the menus.

<i>Menu Title</i>	<i>Command</i>	<i>Accelerator Key</i>
System		Alt-Space
	DOS Shell	Ctrl-Z
	Exit	Alt-X
File		Alt-F
	Open...	Ctrl-O
	Save	Ctrl-S
	Save as...	
	Print...	
	Insert...	Ctrl-I
	Write Block...	Ctrl-W
Edit		Alt-E
	Undelete Line	Ctrl-U
	Restore Line	
	Begin Block	Ctrl-B
	End Block	Ctrl-E
	Hide	Ctrl-H
	Out	Shift-Del, Ctrl-X
	Copy	Ctrl-Ins, Ctrl-C
	Paste	Shift-Ins, Ctrl-V
	Right Shift Block	Shift-Right
	Left Shift Block	Shift-Left
	Set Marker...	Alt-<0..9>

Menu Title	Command	Accelerator Key
Search	Find...	Alt-S
	Translate...	Ctrl-F
	Use Macro...	Ctrl-T
	Repeat	Ctrl-R
View	Select All	Alt-V
	Unselect All	F8
	Line Number...	F9
	Column Number...	Ctrl-G
	Start of Block	
	End of Block	
	Top of Window	Ctrl-Home
	Bottom of Window	Ctrl-End
	Top of File	Ctrl-PgUp
	Bottom of File	Ctrl-PgDn
	Marker...	Ctrl-J
	Previous Position	
Options	Project....	Alt-O
	System...	
	Filters...	
	Editor...	
	Undo limit	
	Marker Display	Ctrl-F4
	Autoindent	Ctrl-F5
	Insert Mode	Ins
	Create Backups	
	Tabs...	
	Set Tabs...	
	Edit Tabs...	
	Ruler Line	Ctrl-F3
	Table Tabs	
	Fixed Tabs	
	Fixed Tab Size...	

Menu Title	Command	Accelerator Key
	Files... Expand Tabs Write Tabs Match Delims Word Defn Move Cursor 43/50 Line Zoom State Save Editor Options	
Project	Automatic Install Automatic Build Automatic Load Install All Build All Compile File	Alt-P Ctrl-F10
Macro	Load... Save... Playback... Record... Edit...	Alt-M Ctrl-P Ctrl-M
Window	Close Navigator Editor Debugger Protocol Statistics NV Browser Resize Zoom On/Off Next Previous	Alt-W Ctrl-Minus F2 F3 F4 F5 F6 F7 Ctrl-F1 Ctrl-F2 Ctrl-N
Help	Keyboard... Editor... Debugger... About...	Alt-H, F1

System Menu

(Alt-Space)

The System pull-down menu is represented by three horizontal bars. This menu contains commands for performing system-level operations.

DOS Shell... (Invoke DOS shell)

(Ctrl-Z)

From the Editor window, selecting DOS Shell gives you access to DOS services or to other programs.

DOS command (<Enter> for shell)

At the prompt box, you can do one of the following:

- Enter one DOS command, just as you would at the DOS prompt, or
- Press Enter to open a DOS shell, which allows you to enter more than one DOS command.

When you open a DOS shell from the LONBUILDER, IDE, the DOS prompt looks like this:

Approximate memory available: 264K
Type EXIT to return...

Microsoft(R) MS-DOS(R) Version 4.01
(C)Copyright Microsoft Corp 1981-1988

LB... C:\LON\LAMPS>

The LONBUILDER software tries to set aside enough memory to allow you to do something in DOS whenever you want. However, if you've been editing several files at once or one very large file, the amount of memory remaining may not be sufficient for your other program, even if you have closed all but a single text window.

When in a DOS shell, do not run any program that starts a TSR (resident) program, as this may interfere with the LONBUILDER software or cause the system to crash.

To return to the LONBUILDER software from the DOS shell, type EXIT at the prompt. You'll be right back where you started.

Exit

(Alt-X)

Selecting **Exit** exits the LONBUILDER software, terminates the LONBUILDER session, and returns you to the DOS prompt.

File Menu

(Alt-F)

The **File** pull-down menu contains commands to perform operations on text files.

Open...(Open a file in a new window)

(Ctrl-O)

From the Editor window, selecting **Open** prompts you for a filename. You are shown the last filename you entered. You can do one of the following:

- Accept it by pressing *Enter*,
- Edit it,
- Enter a new filename or a file mask (for example, *.NC) to display all the files matching the search criteria, or
- Press *Esc* to cancel the prompt.

You can specify any legal filename, including a drive and/or path identifier. (Don't add a .BAK extension to your filenames, because that's the extension the editor appends to backup files.) If you do not specify a filename, the editor text window opens with an UNTITLED file. You can later save it as a named file with the *Save As* command.

If four Editor windows are already open when you select **Open**, you get an error message.

From the Debugger window, selecting **Open** prompts you for a filename. You can specify any legal filename. Press *Esc* to cancel the prompt.

Save (Save file)

(Ctrl-S)

From the editor window, selecting **Save** saves the file in the active window.

If the *Create Backups* option (described under *Options Menu*, later) is on, when existing files are edited and saved, the editor automatically creates a backup copy of the file by renaming the old file and saving the new one under the old name. Backup files are given the .BAK extension.

From the NV Browser window, selecting **Save** saves the current browser list.

Save As... (Save to new filename)

From the editor window, selecting **Save As** prompts you for a filename. You are shown the current filename of the text in the active editor text window. You can do one of the following:

- Accept it by pressing *Enter*,
- Edit it,
- Enter a new filename or a file mask (for example, *.NC) to display all the files matching the search criteria, or
- Press *Esc* to cancel the prompt.

You can specify any legal filename, including a drive and/or path identifier. (Don't add a .BAK extension to your filenames, because that's the extension the editor appends to backup files.) The editor saves the text in the active window to the specified file. This becomes the new name for the file in the window (and in any other windows that display the same file). This command is particularly useful when editing UNTITLED files.

From the NV Browser window, selecting **Save As** prompts you for a file name. You are shown the current browser list file name. You can do one of the following:

- Accept it by pressing *Enter*.
- Edit it
- Press *ESC* to cancel the prompt

Print... (Print file)

Selecting **Print** displays a pop-up menu with the following options:

- **Print file now:** Starts a print job.
- **Name of file:** Selects the file to be printed. If you enter a file mask, you can use the selection bar to select the file from a directory listing.
- **First line:** Starts printing from the specified line number.
- **Last line:** Stops printing at the specified line number.
- **Device:** Selects a file or a printer for the output device.
- **Manual paper feed:** Is YES for manual paper feed on; NO for off.
- **Use form feed:** Is YES to turn form feed on for the printer; NO for off.
- **Page length:** Sets the number of text lines per page.

You can change these parameters in the print setup form. To save them permanently on a system-wide basis, select **Options...Editor** from the main menu bar and click on the **Save Editor** option.

Insert... (Insert file)

(Ctrl-I)

Selecting **Insert** inserts a file into the text at the cursor's current position exactly as if it were copied from another part of the text. The inserted text is marked as a block.

Write Block... (Write block to file)

(Ctrl-W)

Selecting **Write Block** writes the currently marked block to a file. You are first prompted for a filename. If the file already exists, you are asked if you want to overwrite it or append to it. If the file does not exist, a new file is created. The block is left unchanged, and the block markers remain in place. If no block is marked, this command is ignored.

Edit Menu

The **Edit** pull-down menu contains commands for modifying text in the Editor window. Most of the commands on the **Edit** menu work on *blocks* of text. A block is any arbitrarily defined, contiguous unit of text; a block can be as small as a single character or as large as an entire file. You mark a block using one of the following methods:

- Clicking and dragging the mouse, or
- Placing a begin-block marker at the first character in the desired block, and an end-block marker just beyond the last character. (See the *Begin Block* and *End Block* command, later in this section.)

Once marked, the block can be cut, copied, or written to a file.

Although marked blocks are normally highlighted so you can see what you've marked, the block may be *hidden* (or made visible) with the *Hide* command, described later in this section.

Undelete Line

(Ctrl-U)

Selecting Undelete Line restores whole lines deleted with the *Delete Line* command (*Ctrl-D*). To undo your most recent changes to the current line, use the *Restore Line* command, described below. The size of the undelete buffer (where deleted lines are saved) is specified with the *Undo Limit* command, described later under the *Options Menu*.

Restore Line

Selecting Restore Line undoes any changes made to a line of text *as long as you have not left the line*. The line is restored to its previous contents regardless of what changes you have made.

Begin Block (Begin block)

(Ctrl-B)

Selecting Begin Block marks the beginning of a block. The marker itself is not visible on the screen, and the block becomes visible only when the end-block marker is set. You can also use the begin block marker as an extra text marker (see the *Set Marker* command, later in this section) and jump directly to it with the *Start of Block* command, described later under *View Menu*.

End Block (End block)

(Ctrl-E)

Selecting End Block marks the end of a block. Like the begin-block marker, the end block marker is invisible, and the block itself is not displayed unless both markers are set. You can also use the end-block marker as an extra text marker (see the *Set Marker* command, later in this section) and jump directly to it with the *End of Block* command, described later under *View Menu*.

Hide (Hide block)

(Ctrl-H)

Selecting Hide toggles on and off the visual marking of a block. Many of the block manipulation commands work only when the block is being displayed. The block-related cursor movement commands (*Start of Block* and *End of Block*, described later under *View Menu*) work whether the block is hidden or displayed.

Cut (Cut block)

(Shift-Del, Ctrl-X)

Selecting Cut copies the marked and displayed block to the clipboard and deletes the original.

Copy (Copy block)

(Ctrl-Ins, Ctrl-C)

Selecting Copy copies the marked and displayed block to the clipboard. The original block is left unchanged.

Paste (Paste block)**(Shift-Ins, Ctrl-V)**

Selecting Paste inserts the clipboard at the current cursor position. The clipboard is not changed.

Right Shift Block (Shift block right)**(Shift-Right)**

Selecting Right Shift Block shifts the marked block one column right.

Left Shift Block (Shift block left)**(Shift-Left)**

Selecting Left Shift Block shifts the marked block one column left.

Set Marker...**(Alt+<0..9>)**

Selecting Set Marker sets one of the ten text markers at the current position of the cursor. *Alt-0* sets marker 0, *Alt-1* sets marker 1, and so on.

Search Menu**(Alt-S)**

The Search pull-down menu contains commands for finding and replacing character strings in the Editor window.

Find... (Find string)

(Ctrl-F)

Selecting **Find** searches for any string of up to 51 characters. When you enter this command, you are asked for a search string. The last search string entered, if any, or a highlighted block on one line and less than 51 characters is displayed. You can do one of the following:

- Select the displayed search string again by pressing *Enter*
- Edit the displayed search string
- Enter a new search string. *Ctrl-Shift-Ins* can be used to enter control characters (for example, to find a period at the end of a line, you would search for *Enter*, where *Enter* was entered with *Ctrl-Shift-Ins* and *Enter*; to find a */** at the beginning of a line, you would search for *Enter /**,

or

- Press *Esc* to cancel the prompt.

After the search string is entered, you specify your search options. The options you used last, if any, are displayed. You can do one of the following:

- Select the displayed options again by pressing *Enter*
- Edit the displayed options, or
- Enter new options (canceling the old ones).

The following options are available:

- B** Searches backwards from the current cursor position toward the beginning of the file.
- G** Searches globally. The entire file is scanned for the search string, regardless of the current position of the cursor. The search starts at the beginning of the file if searching forwards; at the end if searching backwards.
- L** Limits searches to the currently marked block.
- n** Finds the *n*th occurrence of the string (overridden by the **L** option).

- I** Ignores case; treats all alphabetic characters as if they were uppercase.
- W** Searches for whole words only; skips matching patterns embedded in other words.

If the text contains a target string matching the search string, the target string is highlighted and the cursor is positioned just beyond it.

Translate... (Find and replace string)

(Ctrl-T)

Selecting **Translate** searches for any string of up to 51 characters and replaces the found string with any other string of up to 51 characters. After entering the search string, you are asked to enter the replacement string. The last replacement string entered, if any, is displayed; you may accept it, edit it, or enter a new string as described under *Find*, above.

Finally, you are prompted for options. The options you used last are displayed. You can accept them, edit them, or enter new options (canceling the old ones) as described under *Find*, above. The options available are the same as those for the *Find* command.

Use Macro... (Search and apply macro)

Selecting **Use Macro** searches for any string of up to 51 characters and then applies a macro to it; that is, it moves the cursor to the target string, and executes the commands stored in the macro.

The *Use Macro* command accepts search strings in the same way as the *Find* command. After entering the search string, you are asked to select a macro from the displayed list of all the macros you've defined. Move the selection bar with the *Up* and *Down* arrow keys, and select your macro by pressing *Enter*. You can cancel the operation by pressing *Esc*.

Finally, you are prompted for options. The options you used last are displayed. You can accept them, edit them, or enter new options (canceling the old ones) as described under *Find*, earlier. The options available are the same as those for the *Find* command.

The screen is not updated while a *Use Macro* command is operating.

Because the *Use Macro* command can be invoked *inside a macro*, it is possible to create an almost unlimited array of special commands. Several examples of such macros are predefined in the LBE.MAC file in the \LB directory. Studying these macros may give you ideas for your own macros. To look at them, open the Macro pull-down menu, select the Load command to load the LBE.MAC file, and select the Edit command from the Macro menu. Refer to the *Macro Menu*, later, for details.

Repeat (Find next)

(Ctrl-R)

Selecting Repeat repeats the most recent search command. If the most recent search is a *Find*, the same search string and options are repeated; if the most recent search is a *Translate*, the replacement string is reused as well.

View Menu

(Alt-V)

The View pull-down menu contains extended cursor movement commands in the Editor window and two object selection commands in the Navigator window.

Select All (Select all objects in an object list) (F8)

Selecting Select All selects all objects in an object list on the Navigator window.

Unselect All (Unselect any selected objects) (F9)

Selecting Unselect All unselects any selected objects in an object list on the Navigator window.

Line Number... (Go to line) (Ctrl-G)

Selecting Line Number prompts for a line number and moves the cursor to the specified line. Any positive integer value in the range 1 to 32,767 is valid. If the value is preceded by a plus (+) or minus (-) sign, the target line number is calculated relative to the current line. Line numbers are counted from the beginning of the file.

Column Number... (Go to column)

Selecting Column Number prompts for a column number and moves the cursor to the specified column of the current line. Any positive integer value in the range 1 to 999 is valid. If the value is preceded by a plus (+) or minus (-) sign, the target column number is calculated relative to the current column.

Start of Block (Top of block)

Selecting Start of Block moves the cursor to the position of the block-begin marker set with *Ctrl-B*. The command works even if the block is hidden or the end-block marker is not set.

End of Block (Bottom of block)

Selecting End of Block moves the cursor to the position of the block-end marker set with *Ctrl-E*. The command works even if the block is hidden or the begin-block marker is not set.

Top of Window

(Ctrl-Home)

Selecting Top of Window moves the cursor to the first line displayed in the active window. The cursor remains in the same column.

Bottom of Window

(Ctrl-End)

Selecting **Bottom of Window** moves the cursor to the last line displayed in the active window. The cursor remains in the same column.

Top of File (Beginning of file)

(Ctrl-PgUp)

Selecting **Top of File** moves the cursor to the first character in the file.

Bottom of File (End of file)

(Ctrl-PgDn)

Selecting **End of File** moves the cursor just beyond the last character in the file.

Marker... (Jump to marker)(Ctrl-J or Shift-Alt-0..Shift-Alt-9)

Selecting **Marker** moves the cursor to one of the ten text markers created with the *Set Marker* command. A dialog box is displayed, showing the marker number, the file it is in, and the number of the line containing the marker. If a given marker has not been set, **Not Set** is displayed instead. You may select a marker either by pressing its number or by moving the selection bar to it and pressing *Enter*. To jump directly to a marker without viewing the marker dialog box, use *Shift-Alt-<0..9>*. *Shift-Alt-0* jumps to marker 0, *Shift-Alt-1* jumps to marker 1, and so on. If the specified marker has not been set, the cursor is not moved.

Previous Position

Selecting **Previous Position** moves to the last cursor position. This command is useful to move back to the previous position after a *Find* or *Translate* operation.

Options Menu

(Alt-O)

The Options pull-down menu contains commands that open dialog boxes for setting options.

Project... (Set project parameters)

Selecting **Project** opens the project parameter configuration window. This window is described under *Configuring LONBUILDER Project Parameters* in Chapter 4. This command is not available from the Editor window.

System... (Set system parameters)

Selecting **System** opens the system parameter configuration window. This window is described under *Configuring LONBUILDER System Parameters* in Chapter 4. This command is not available from the Editor window.

Filter... (Set protocol filters)

Selecting **Filter** opens the protocol filters configuration window. This window is described under *Defining the Packet Log Filter* in Chapter 12. This command is available from the Protocol Analyzer and Statistics windows only.

Editor... (Set editor options)

Selecting **Editor** opens a menu with commands for setting the *Undo Limit*, *Marker Display*, *Autoindent*, *Insert Mode*, *Create Backups*, *Tabs*, and *Files* editor options. These commands are available from the **Editor** window only.

Undo Limit (Set undo limit)

Selecting **Undo Limit** sets the size of the undelete buffer, which stores lines deleted with the *Delete Line* (*Ctrl-D*) command. The default value is 20 lines.

Marker Display (Toggle marker display) (Ctrl-F4)

Selecting **Marker Display** hides or reveals all text markers. Setting a new text marker automatically turns on marker display if it was previously turned off.

Autoindent (Toggle autoindent) (Ctrl-F3)

Selecting **Autoindent** toggles the autoindent on and off. When *Autoindent* is on, the *Enter* key moves the cursor to the next row and to the same column as the first nonblank character on the previous line.

Insert Mode (Toggle insert mode) (Ins)

Selecting **Insert Mode** toggles the Insert or Replace mode. In Insert mode, text to the right of the cursor is moved to the right as new text is entered. In Replace mode, any text below the cursor is overwritten when new text is entered. The cursor's size is itself an indication of which mode you are in: a tall cursor indicates Insert, while a thin cursor indicates Replace.

Create Backups (Sets backup file creation)

Selecting **Create Backups** sets the creation of a backup file every time an existing file is saved. The backup file has a .BAK suffix appended to it.

Tabs... (Set tab options)

Selecting **Tabs** opens a menu with commands for setting and changing tabs.

Set Tabs... (Set rule line)

Selecting **Set Tabs** sets tab stops based on the positions of the words on the current line. This is useful when entering information in tables that must all look alike.

Edit Tabs... (Edit ruler line)

Selecting **Edit Tabs** allows you to alter the locations of the tab stops. Pressing the space bar inserts or removes tab stops at the position of the cursor. Pressing *Esc* returns the tab stops to their previous settings. You can also add new stops by pressing *Ins* and delete existing ones by pressing *Del*. The *Left* and *Right arrow* keys move the cursor along the tab line; *Tab* moves it to the next stop; *Home* moves it to the first stop; and *End* moves it to the last one. When you are finished, you can return to the text window by pressing *Enter*.

Ruler Line (Toggle ruler line)

Selecting **Ruler Line** turns on the ruler line for the current window. The ruler line shows you the position of all tab stops.

This command turns on the ruler line, if necessary, and leaves it on.

Turning on the ruler line automatically turns off table tabs.

Table Tabs (Toggle table tabs)

Selecting **Table Tabs** changes the tabs with every line: The first letter in each word on the previous line is treated as a tab stop.

When table tabs are not in effect, the tab stops (normally) start at column 5 and occur every 4 columns thereafter. By default, table tabs are off. When table tabs are in use, the tab stops are determined by the locations of the words on the previous line; the first character in each word represents a tab stop. To change the locations of the stops when not using table tabs, use the *Fixed Tab Size*, *Edit Tabs*, or *Set Tabs* commands all described in this section.

Fixed Tabs (Default tabs)

Selecting **Fixed Tabs** cancels any special tab settings specified with the *Set Tabs* and *Edit Tabs* commands, and restores evenly spaced tab stops based on the current fixed tab size.

Fixed Tab Size... (Set fixed tab size)

Selecting **Fixed Tab Size** sets the default fixed tab size to be used when fixed tabs are on and table tabs are off. This value is also used when expanding tab characters if the tab expansion option is on. See the discussion of the *Expand Tabs* command, below. The default tab size is 4.

Files... (Set edit file options)

Selecting **Files** opens a menu with commands for modifying text files when they are opened and saved.

Expand Tabs (Toggle Tab Expansion)(Ctrl-F4)

Selecting **Expand Tabs** expands tab characters to spaces when reading in files. The assumed size of the tabs is specified with the *Fixed Tab Size* command, described earlier. Spaces may be converted back to tabs when output is written. (See the *Write Tabs* command, below.) The default value is to expand tabs.

Write Tabs (Toggle Tab Compression)

Selecting **Write Tabs** toggles tab compression on and off. When *Write Tabs* is on, the editor converts strings of blanks to tabs when a file is written to disk. This saves disk space for indented programs. The current fixed tab spacing is used when converting blanks into tabs. The default for this toggle is off (spaces are not converted to tabs).

Match Delims (Toggle Automatic Delimiter Matching)

Selecting **Match Delims** toggles automatic matching of delimiters on and off. When *Match Delims* is on, whenever a closing delimiter ")", "]", or ")" is entered, the editor will momentarily position to the corresponding opening delimiter and then return to the current position. Entering another character or command will cause the cursor to return immediately.

Word Defn (Toggle Definition of Word for Cursor Movement)

Selecting **Word Defn** toggles the definition of a "word" for the editor move word left and word right commands.

Selecting **Word** causes word left and right commands to move to the beginning of the next group of alpha-numeric characters. Selecting **LEX** causes word left and word right commands to move to the beginning of the next lexical group of characters.

e A-25 Addition

dd the following editor options (before "Save Editor Options"):

3/50 Line

If OFF is selected, the screen will display 25 lines by 80 columns. If ON is selected, the screen will display 43 lines by 80 columns if you are using an EGA display, or 50 lines by 80 columns if you are using a VGA display. The default is OFF.

com State

If OFF is selected, when a file is opened it will share the screen with any other files that are currently open (up to the maximum number). If ON is selected, when a file is opened it will automatically zoom to fill the whole screen. The default is OFF.

Move Cursor (Toggle Automatic Cursor Movement)

Selecting Move Cursor toggles automatic cursor movement on and off. When Move Cursor is ON, selecting a new window using the mouse also causes the cursor to be moved to the point where the mouse was clicked. When OFF, the cursor remains at the same position as the last time the window was active.

Save Editor Options (Save editor defaults)

Selecting Save Editor Options saves the current editor configuration settings (specified by the *Options Editor* commands) as the defaults.

Project Menu

(Alt-P)

The Project pull-down menu contains commands that operate on the current project.

Automatic Install (Install Target Hardware)

Selecting Automatic Install installs the target hardware of all nodes defined in the object database that require it.

Automatic Build (Build current project)

(Ctrl-F10)

Selecting **Automatic Build** recompiles and relinks the project's NEURON C and include files that require it. If the **Stop After Compile** project parameter is off, the *Automatic Build* command connects message tags and network variables. If the **Load After Build** project parameter is on, the *Automatic Build* command loads the application images and configures the network images after compiling and linking, as needed. If the **Start After Load** project configuration parameter is on, the *Automatic Build* command starts all newly loaded or configured nodes.

Automatic Load (Load current project)

Selecting this command loads existing application images to the project's nodes if the node's application image is out of date. If you select this command, any necessary compiles, links, or binds will be done prior to loading. This command is identical to an *Automatic Build* with the **Load After Build** option set.

Install All (Install all of the current project)

Selecting **Install All** installs the target hardware of all nodes defined in the object database.

Build All (Build all of the current project)

Selecting **Build All** recompiles and relinks all program files. If the **Stop After Compile** project parameter is off, the *Build All* command connects message tags and network variables. If the **Load After Build** project parameter is on, the *Build All* command loads and configures all nodes after connecting. If the **Start after Load** project configuration parameter is on, the **Automatic Build** command starts all newly loaded or configured nodes.

Compile File (Compile current file)

Selecting **Compile** from the Editor window compiles the source file in the active text window.

Macro Menu

(Alt-M)

The **Macro** pull-down menu contains commands to perform operations on macros in the Editor window.

Macros simplify repetitive tasks by storing sequences of commands and keystrokes. The editor allows you to record, edit, save, and load up to ten macros, each containing up to 255 keystrokes. You can have as many macro files as your disks can hold, but only one file of ten macros can be kept in memory at a time. (Refer to the *Use Macro* command, described earlier in the *Search Menu* section.)

Load... (Load macros from disk)

Selecting **Load** loads a file of previously saved macros. If the file you specify does not exist, an error message is displayed.

Save... (Save macros to disk)

Selecting **Save** saves the current macros to the file you specify.

Playback... (Play back macro)(Ctrl-P or Alt-F1...Alt-F10)

Selecting **Play Back** prompts you for a choice, then plays back the macro you select. Macros can be executed without prompting with *Alt-F1* through *Alt-F10*.

Record... (Toggle macro record) (Ctrl-M)

Selecting **Record** turns macro recording on or off. When macro recording is on, you are asked to select a macro slot to store the recording in. All subsequent keystrokes (up to the 255-keystroke limit) are saved until macro recording is turned off again. (Keystrokes used to turn *Record* on and off are not saved.)

Edit... (Edit macro definition)

Selecting **Edit** allows you to edit previously saved macros. The editor has a special editing module designed specifically for working with macros. Special keys (function keys, cursor keys, and so on) are represented by highlighted, abbreviated forms of their names, for example, *Esc*, *Ctrl-N*, and *Enter*. Regular keystrokes are shown as normal characters without highlighting.

When you enter this command, you are asked to select a macro to edit (and optionally a name for the macro). Then you enter the macro editor itself. Several keys serve special purposes here:

- Cursor keys behave as usual, moving the cursor so you can select a particular keystroke to change
- *Backspace* deletes a single keystroke, just as it does in the regular editor
- *Ctrl-Backspace* deletes an entire macro
- *Esc* restores the macro to its previous state and exits the macro editor
- *Enter* takes you out of the macro editor
- *Scroll Lock* acts as a toggle, turning literal interpretation of keystrokes on and off. For example, to assign a macro to *Enter* or *Backspace*, which normally serve a special function, you first press *Scroll Lock*. Pressing it again restores all special keys to their previous functions.

Window Menu

(Alt-W)

The Window pull-down menu contains commands that operate on windows.

Close (Close window)

(Ctrl-Minus)

Selecting Close in the Editor window closes the current text window. If the file has not been edited since the last time it was saved, the window is closed immediately. If it has been edited, you are asked whether you want to save the file or close the window without saving.

Navigator (Change to Navigator window)

(F2)

Selecting Navigator changes to the Navigator window.

Editor (Change to Editor window)

(F3)

Selecting Editor changes to the Editor window.

Debugger (Change to Debugger window)

(F4)

Selecting Debugger changes to the Debugger window.

Protocol (Change to Protocol window)

(F5)

Selecting Protocol changes to the Protocol Analyzer window.

Statistics (Change to Network Statistics window) (F6)

Selecting **Statistics** changes to the **Network Statistics** window.

NV Browser (F7)

Selecting **NV Browser** changes to the **Network Variable (NV) Browser** window.

Go To... (Go to window)

Selecting **Go To** in the **Editor** window displays a list of the open text windows in a dialog box. You select the window to which you want to go, and the cursor is moved to the indicated window. If only one window is open, an error message is displayed.

Selecting **Go To** in the **Debugger** window displays a list of alternate emulators that may be debugged. Select the emulator you want to go to. The program for the selected emulator will be displayed in the **Debugger** window.

Resize (Resize active window) (Ctrl-F1)

Selecting **Resize** in the **Editor** window changes the size of the active text window. You can adjust the size by pressing the *Up* and *Down* arrow keys. When you are finished, pressing *Enter* or *Esc* returns you to the text window.

Zoom (Toggle zoom for current window) (Ctrl-F2)

Selecting **Zoom** in the Editor window toggles between zoomed and unzoomed text windows. Zoomed windows fill the entire screen with the active window, hiding the other text windows. Unzoomed windows appear on screen at the same time. If you change windows while zooming, the window you change to is zoomed as well.

Next (Next Window) (Ctrl-N)

Selecting **Next** in the Editor window makes the next text window the active window.

Selecting **Next** in the Debugger window makes the next evaluation window the active window.

Previous (Previous window)

Selecting **Previous** in the Editor window makes the previous text window the active window.

■ Page A-32 Addition

Add the following text to the end of "Previous (Previous Window)":

Selecting **Previous** in the Debugger window makes the previous evaluation window the active window.

Help Menu

(Alt-H or F1)

The Help pull-down menu contains command options that offer help information.

Keyboard

Selecting Keyboard lists the keyboard commands.

Editor

The Editor help command is only available from the Editor window.

From the Editor window, selecting Editor displays a menu of help items pertinent to the editor.

Show Help (Help summary)

Selecting Show Help displays a brief description of the editor help system and how it works.

Help and Status (Status help)

Selecting Help and Status provides help on the editor help system itself and on the editor *About* command, described earlier under *System Menu*.

Cursor Movement (Cursor help)

Selecting Cursor Movement provides help on the basic editor cursor commands.

Quick Movement (Quick movement help)

Selecting Quick Movement provides help on the quick editor cursor movement commands.

Insert and Delete (Delete help)

Selecting **Insert** and **Delete** provides help on the editor text insertion and deletion commands.

System (System help)

Selecting **System** provides help on the commands that give access to system-level functions (those described under *System Menu*, earlier).

Files (File help)

Selecting **Files** provides help on the editor file commands.

Edit (Edit help)

Selecting **Edit** provides help on the editor edit commands.

Search and Replace (Find and translate help)

Selecting **Search** and **Replace** provides help on the editor *Find*, *Translate*, and *Use Macro* commands.

View (View help)

Selecting **View** provides help on the editor view commands.

Tabs (Tab help)

Selecting **Tabs** provides help on the editor tab commands.

Options (Options help)

Selecting **Option** provides help on the editor commands that change default settings or that toggle particular options on and off.

Project (Project help)

Selecting **Project** provides help on the project commands that can be accessed from the Editor.

Macros (Macro help)

Selecting **Macros** provides help on the editor macro commands.

Windows (Window help)

Selecting **Windows** provides help on the window commands.

Function keys (Function key help)

Selecting **Function Keys** provides a quick reference list, showing all of the commands assigned to function keys.

Debugger

Selecting **Debugger** provides help on the Debugger.

About... (Show system info)

From the editor Help menu, selecting About displays editor, current file, and system statistics, including the

- Editor version number
- Full name of the file being edited and if it has or has not been modified
- Number of lines, bytes, and words in the file
- Current time and date
- Amount of memory remaining
- Version of DOS being used
- Amount of space remaining on the currently logged drive
- Current drive and directory.

Because some of the file statistics (notably the word count) can take a long time to compute, you may want to skip them by pressing any key while they are being updated. This tells the editor to stop what it is doing and jump ahead to the system information section.

From any LONBUILDER window other than the Editor window, selecting About displays the amount of memory remaining, as well as copyright and version information for the LONBUILDER software.

B

Editor Keyboard Summary

This appendix summarizes the keystrokes that invoke LONBUILDER Editor commands.

Keystroke	Command
Backspace	Delete Character Left
Del	Delete Character Right or Block
Down Arrow	Line Down
End	End of Line
Enter	Insert New Line
Esc	Cancel
Home	Beginning of Line
Insert	Toggle Insert/Replace Mode
Left Arrow	Character Left
PgUp	Page Up
PgDn	Page Down
Right Arrow	Character Right
Tab	Tab Right
Up Arrow	Line Up
Ctrl-Backspace	Delete Word Left
Ctrl-Del	Delete Word Right
Ctrl-Down	Scroll Window Down
Ctrl-End	Bottom of Window
Ctrl-Enter	Open New Line
Ctrl-Home	Top of Window
Ctrl-Ins	Copy Block
Ctrl-Left	Word Left
Ctrl-Minus	Close Window
Ctrl-PgDn	Bottom of File
Ctrl-PgUp	Top of File
Ctrl-Right	Word Right
Ctrl-Up	Scroll Window Up

Keystroke	Command
Shift-Del	Cut Block
Shift-Down	Down to Equal Indent
Shift-Ins	Paste Block
Shift-Left	Shift Block Left
Shift-Right	Shift Block Right
Shift-Tab	Tab Left
Shift-Up	Up to Equal Indent
Ctrl-Shift-Ins	Insert Control Character
Ctrl-Shift-Right	Select Word
Ctrl-B	Begin Block
Ctrl-C	Copy Block
Ctrl-D	Delete Line
Ctrl-E	End Block
Ctrl-F	Find
Ctrl-G	Go To Line
Ctrl-H	Hide Block
Ctrl-I	Insert File
Ctrl-J	Jump to Marker
Ctrl-K	Delete to End of Line
Ctrl-L	Line Mark
Ctrl-M	Toggle Macro Record
Ctrl-N	Next Window
Ctrl-O	Open File
Ctrl-P	Playback Macro
Ctrl-R	Repeat Search
Ctrl-S	Save File
Ctrl-T	Translate
Ctrl-U	Undelete Line
Ctrl-V	Paste Block
Ctrl-W	Write Block
Ctrl-X	Cut Block
Ctrl-Z	DOS Shell
Ctrl-}	Show matching left delimiter
Ctrl-{	Show matching right delimiter

<i>Keystroke</i>	<i>Command</i>
Alt-Space	System Menu
Alt-E	Edit Menu
Alt-F	File Menu
Alt-H	Help Menu
Alt-M	Macro Menu
Alt-O	Options Menu
Alt-P	Project Menu
Alt-S	Search Menu
Alt-V	View Menu
Alt-W	Window Menu
Alt-X	Exit
Alt-<F1..F10>	Playback Macro 1..10
Alt-<0..9>	Set Marker 0..9
Shift-Alt-<0..9>	Jump to Marker 0..9
F1	Help
F2	Navigator Window
F3	Editor Window
F4	Debugger Window
F5	Protocol Window
F6	Statistics Window
F7	NV Browser Window
F8	Select All
F9	Unselect All
F10	Menu Bar
Ctrl-F1	Resize Window
Ctrl-F2	Toggle Zoom
Ctrl-F3	Toggle Ruler Line
Ctrl-F4	Toggle Marker Display
Ctrl-F5	Toggle Autoindent
Ctrl-F6	Change Block to Lower Case
Ctrl-F7	Change Block to Upper Case
Ctrl-F8	Next Error or Warning
Ctrl-F9	Next Error
Ctrl-F10	Automatic Build

C

Sample Memory Map

This appendix shows a portion of a detailed memory map generated when the Generate Link Map build option is on. See *Specifying Build Options*, in chapter 4, for details on specifying this option.

Sample NEURON 3150 CHIP Memory Map

This section shows pages 1 and 2 of a sample memory map for a NEURON 3150 CHIP.

Page 1: NEURON 3150 CHIP Memory Map Sample

NEURON CHIP (R) Linker V2.1.002 copyright (c) 1989,1992 Echelon Corp
Page 1 Fri Jan 31 16:57:32 1992 PROD\OPTE7C3E.nxr

LIST OF MEMORY AREAS

* AREA *		* ALLOCATED *			* UNUSED *		
		From	To	Len	From	To	Len
1	SYSTEM IMAGE ROM	0000	3FFF	4000	-RESERVED-		
2	Offchip ROM	4000	47FF	0800	47AB	47FF	0055
	Offchip ROM (Pool)	-	AVAILABLE	-	4800	4FFF	0800
3	Offchip EEPROM	(Pool)	-	AVAILABLE	-	8000	89FF
	Offchip RAM (Pool)	-	AVAILABLE	-	9200	9DFF	0C00
5	OnChip RAM (Sys)	E800	EA2F	0230	-	NONE	-
	(cont'd)	EA70	EC48	01D9	-	NONE	-
	OnChip RAMFAR	EA30	EA6F	0040	EA30	EA6F	0040
	OnChip RAM (Pool)	-	AVAILABLE	-	EC09	EFED	03E5
6	OnChip RAMNEAR	EFEE	FFFF	0012	-	NONE	-
7	OnChip EECODE	F000	F0A6	00A7	-	NONE	-
8	OnChip EEPROM (Pool)	-	AVAILABLE	-	F0A7	F1EA	0144
	OnChip EEPROM (Map)	F1EB	F1F3	0009	-	NONE	-
	OnChip EEPROM (Sys)	F1F4	F1F4	0001	-	NONE	-
9	OnChip EENEAR	F1F5	F1FE	000A	-	NONE	-
10	OnChip EEPROM (Sys)	F1FF	F1FF	0001	-	NONE	-

End of Memory Area List

This memory map lists the areas of memory that were allocated and tells if the allocation was used. Off-chip memory is allocated in 256-byte pages; on-chip memory is allocated in bytes.

- 1 0000 to 3FFF off-chip ROM was reserved for system firmware.
- 2 4000 to 47FF off-chip ROM was used for application code and data; 47AB to 47FF of that allocated area was unused; 4800 to 4FFF pool area was available to be allocated and was not used.
- 3 8000 to 89FF off-chip EEPROM pool area was available to be allocated and was not used.
- 4 9200 to 9DFF off-chip RAM pool area was available to be allocated and was not used.
- 5 Two areas of on-chip RAM were allocated and used for system data: E800 to E82F and, EA70 to EC48. The area from EA30 to EA6F is available for RAMFAR, but was not used. EC09 to EFED pool area was available to be allocated and was not used.
- 6 EFEE to EFFF on-chip RAMNEAR was allocated for system application tables and code and was all used.
- 7 E600 to E6A6 on-chip EECODE was allocated for application variables and was all used.
- 8 E6A7 to E7EA pool area was available to be allocated and was not used; E7EB to E7F3 was allocated for the map and was all used; E7F4 to E7F4 was allocated for system variables and was all used.
- 9 E7F5 to E7FE was allocated for application variables and was all used.
- 10 E7FF to E7FF was allocated for system variables and was all used.

Page 2: NEURON 3150 CHIP Memory Map Sample

NEURON CHIP (R) Linker V2.1.002 copyright (c) 1989,1992 Echelon Corp
Page 2 Fri Jan 31 16:57:32 1992 PROD\OPTE7C3E.nxr

The Link Memory Usage Statistics are also placed in the BUILD.LOG file, if the Output Link Summary build option is on. See Specifying Build Options, in chapter 4, for details.

Link Memory Usage Statistics:

ROM Usage:

Application Code & Constant Data	1881 bytes
Library Code and Constant Data	0 bytes
Self-Identification Data	82 bytes

Total ROM Usage	1963 bytes

EEPROM Usage (not necessarily in order of physical layout):

System Data & Parameters	73 bytes
Domain & Address Tables	105 bytes
Network Variable Config Tables	0 bytes
Application EEPROM Variables	10 bytes
Library EEPROM Variables	0 bytes
Application Code & Constant Data	0 bytes
Library Code & Constant Data	0 bytes

Total EEPROM Usage	188 bytes

RAM Usage(not necessarily in order of physical layout):

System Data & Parameters	567 bytes
Transaction Control Blocks	24 bytes
Appl Timers & I/O Change Events	0 bytes
Network & Application Buffers	424 bytes
Application RAM Variables	18 bytes
Library RAM Variables	0 bytes

Total RAM Usage	1033 bytes

Linked for 3150:

Memory Map (9 bytes of System EEPROM) is 3150 specific.

Some system RAM (109 bytes) is 3150 specific.

End of Link Statistics

Sample NEURON 3120 CHIP Memory Map

This section shows pages 1 and 2 of a sample memory map for a NEURON 3120 CHIP.

Page 1: NEURON 3120 CHIP Memory Map Sample

NEURON CHIP (R) Linker V2.1.002 copyright (c) 1989,1992 Echelon Corp
Page 1 Fri Jan 31 16:57:32 1992 PROD\OPTE7C3E.nxr

LIST OF MEMORY AREAS

* AREA *	* ALLOCATED *	* UNUSED *
	From To Len	From To Len
SYSTEM IMAGE ROM	0000 27FF 2800	-RESERVED-
OnChip RAM (Sys)	EC00 EF41 0342	- NONE -
OnChip RAM (Pool)	- AVAILABLE -	EF46 EFF9 00B4
OnChip RAMFAR	EF42 EF45 0004	- NONE -
OnChip RAMNEAR	EFF8 EFEF 0008	- NONE -
OnChip EECODE	F000 F0DF 00E0	- NONE -
OnChip EEPROM (Pool)	- AVAILABLE -	F0E0 F1FD 011E
OnChip EEPROM (Sys)	F1F8 F1F8 0001	- NONE -
OnChip EENEAR	F1F9 F1FE 0006	- NONE -
OnChip EEPROM (Sys)	F1FF F1FF 0001	- NONE -

End of Memory Area List

Page 2: NEURON 3120 CHIP Memory Map Sample

NEURON CHIP (R) Linker V2.1.002 copyright (c) 1989,1992 Echelon Corp
Page 2 Fri Jan 31 16:57:32 1992 PROD\OPTE7C3E.nxr

Link Memory Usage Statistics:

EEPROM Usage(not necessarily in order of physical layout):

System Data & Parameters	64 bytes
Domain & Address Tables	105 bytes
Network Variable Config Tables	3 bytes
Application EEPROM Variables	6 bytes
Library EEPROM Variables	0 bytes
Application Code & Constant Data	41 bytes
Library Code & Constant Data	5 bytes
Self-Identification Data	8 bytes

Total EEPROM Usage	232 bytes

RAM Usage(not necessarily in order of physical layout):

System Data & Parameters	454 bytes
Transaction Control Blocks	62 bytes
Appl Timers & I/O Change Events	18 bytes
Network & Application Buffers	300 bytes
Application RAM Variables	12 bytes
Library RAM Variables	0 bytes

Total RAM Usage	846 bytes

Linked for 3120:

Remaining EEPROM = 286 bytes Remaining RAM = 178 bytes.

End of Link Statistics

D

Utilities

This appendix describes the various utility programs included with the LONBUILDER software.

HEXCNVRT

Convert hex data files between Intel Hex and Motorola S-record formats.

Usage:

HEXCNVRT <switches> <in file> <out file> where <switches> are optional and may be one or more of:

- a append to existing file
- ic ignore existing checksums
- nt do not generate a terminator record
- nc do not convert the record format

HEXCNVRT takes as arguments an input file and an output file, with optional switches. In all cases, all the data records from the input file are copied to the output file. If no switches are used, the checksums are validated and the record format of the output file will be the reverse of the input file: Intel Hex is converted to Motorola S-record, and Motorola S-record is converted to Intel Hex. If the output file already exist, it will be overwritten. A terminator record will be written at the end of the file. The switches may be used to modify the default behavior. For example, if you wish to concatenate several (N) files and place a single terminator record at the end, use the -nt switch when converting files 1 through N-1, and use the -a switch when converting files 2 through N. If you do not want the format converted, use the -nc switch for each file.

Another use for HEXCNVRT is to regenerate the record checksums if you need to edit the hex data. Using both the -ic and the -nc switches will ignore the old (incorrect) checksums, calculate new checksums, and write the records out in the same format. A brief help message is printed if this command is entered with no arguments.

LBCHECK

Check the available memory in your PC.

Usage: LBCHECK

LBCHECK will display the available conventional, extended, and expanded memory that it detects in your PC. Chapter 4 explains how to specify overlay memory use for LONBUILDER. Chapter 3 of the *LONBUILDER Startup and Hardware Guide* discusses setting up the memory in your PC.

LBDBCHK

Check a LONBUILDER database for consistency.

Usage: LBDBCHK <project dir> <system dir>

LBDBCHK takes as arguments the full pathnames, including drives, of the project and system directories that require a database check. The internal consistency of the database is checked. The output is somewhat verbose, but only the last line is of real interest. It shows the number of errors encountered in scanning the database. If this number is not zero, you have a database problem and should try running LBDBFIX. The consistency checking only validates record and key relationships, not record contents. If no errors are reported, this does not guarantee that there are no errors in the actual data contained in database.

A brief help message is printed if this command is entered with less than two arguments.

LBDBFIX

Rebuild the record keys for a LONBUILDER database.

Usage: LBDBFIX <project dir> <system dir>

LBDBFIX takes as arguments the full pathnames, including drives, of the project and system directories that require a database key rebuild. If LBDBCHK reports errors, you should try running LBDBFIX. This will rebuild all of the database key files, which will fix inconsistent key references. However, there are other types of errors reported by LBDBCHK that cannot be repaired by LBDBFIX. In addition, it cannot fix errors in the data contained in the database.

LBDBCHK may also be used to compress key files after a large number of records have been created and deleted.

A brief help message is printed if this command is entered with less than two arguments.

LBTEST

Test the LONBUILDER development station hardware.

Usage: LBTEST

LBTEST is an interactive program that performs various tests on the LONBUILDER development station hardware. It is fully documented in Chapter 6 of the LONBUILDER Startup and Hardware Guide.

■ Page D-4 Addition

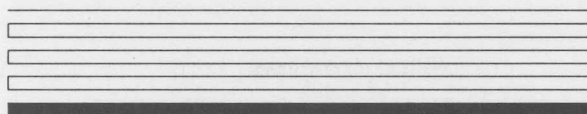
Add the following utility description:

XIF3TO2

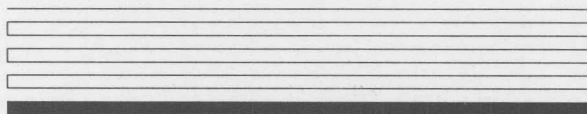
Convert a LONBUILDER external interface file (.XIF) from version 3 format to version 2 format.

Usage: XIF3TO2 <input XIF file> <output XIF file>

The input file must be version 3 format (the default exported by LONBUILDER 2.2), and the output file will be version 2 format. This conversion is needed to use external interface files exported from LONBUILDER 2.2 with a product based on the LONMANAGER API for DOS release 2.11 or the LONMANAGER API for Windows release 2.11.



Index



Index

.BIF file 7-12, 7-15, 7-22, 7-23
.HC 6-20, 7-14
.NC file 6-20, 7-4
.NEI file 7-12, 7-32, 7-35
.NO file 6-20, 7-12, 7-15, 7-22, 7-23
.NRI file 7-12, 7-13, 7-32, 7-33, 7-34
.NXE file 7-12, 7-14, 7-15, 7-16, 7-17
.XIF file 6-20, 6-34, 6-39, 7-1, 7-12, 7-14, 7-15, 7-16, 7-21

A

accelerator keys 3-4, 3-5, Appendix A
about - show system info A-36
autoindent A-21
automatic build A-26
automatic install A-25
automatic load A-26
begin block A-11
bottom of block A-18
bottom of file A-19
bottom of window A-19
build all A-27

A

accelerator keys (continued)
change to editor window A-30
close A-30
column number A-18
compile file A-27
copy block A-12
create back-ups A-22
cut block A-12
cursor movement A-33
debugger A-30, A-35
DOS shell A-5
edit help A-34
edit macro definition A-29
edit menu A-2, A-10
edit tabs A-22
editor A-21, A-33
end block A-12
exit A-6
expand tabs A-24
file help A-34
file menu A-2, A-7
files A-23
filter A-20
find string A-14
fixed tabs A-23
fixed tab size A-23
function keys help A-35
go to A-31
help and status A-33
help menu A-4, A-33
hide block A-12
insert and delete A-34
insert file A-10
insert mode A-21
install all A-26
keyboard A-33
left shift block A-13
line number A-17
load A-28
open window A-7
options menu A-3, A-20

A

accelerator keys (continued)

- protocol A-30
- macro menu A-4, A-27
- macros help A-35
- marker A-19
- marker display A-21
- match delims A-24
- move cursor A-25
- navigator A-30
- next window A-32
- nv browser A-31
- options help A-34
- paste block A-13
- playback A-28
- previous position A-20
- previous window A-32
- print A-9
- project help A-35
- project menu A-4, A-25
- project parameters A-20
- quick movement A-33
- record A-28
- repeat, find next A-16
- resize A-31
- restore line A-11
- right shift block A-13
- ruler line A-22
- save as A-8
- save editor options A-25
- save file A-8
- save macros A-28
- search and replace A-34
- search menu A-2, A-13
- select all A-17
- set marker A-13
- set tabs A-22
- show help A-33
- start of block A-18
- statistics A-31
- system help A-34
- system menu A-3, A-5

A

accelerator keys (continued)

protocol A-30

macro menu A-4, A-27

system parameters A-20

tab help A-34

table tabs A-23

tabs A-22

top of file A-19

top of window A-18

translate A-15

undelete line A-11

undo limit A-21

unselect all A-17

use macro A-15

view help A-34

view menu A-3, A-17

word defn A-24

window menu A-4, A-30

windows help A-35

write block A-10

write tabs A-24

zoom A-32

actuators 1-2

application definition 2-8

application development cycle 2-2, 2-3, 2-4

application directories 4-4

application image create window 6-21

app image name field 6-21

app image origin field 6-22

application interface board 1-10

application 6-2 *see also* node (s)

application image 2-2, 2-3, 6-2, 6-20, 7-2

application image create window 6-21

application node selector window 6-3

A

- application (continued)
 - building application nodes 7-2
 - automatic build command 7-3
 - automatic load command 7-3
 - build all command 7-3
 - compiles remaining 7-4
 - links remaining 7-4
 - loads remaining 7-4
 - project manager window 7-4
 - channel parameters 6-4
 - default_channel 6-4
 - creating node specifications 6-2, 6-3
 - firmware *see* firmware
 - hardware properties 6-3 *see also* hardware properties
 - image 6-4, 6-20 *see also* image
 - installing target hardware 6-26
 - network image 6-2
 - node specifications 6-4, 6-23
 - node specification create window 6-24
 - off-line 13-10
 - on-line 13-10
 - object database 6-4, 6-5
 - objects 6-30
 - deleting 6-31
 - modifying 6-30
 - target hardware 6-3 *see also* target hardware
- application node selector window 6-3
- application programs 2-6, 3-2
 - creating 2-6
 - testing 2-7
- application programming tools 1-11
 - see also* integrated development environment
 - compiler 1-11, 1-12
 - debugger 1-11
 - editor 1-11
 - project manager 1-11, 1-12
 - source level debugger 1-11

A

- application source code 1-8
- application windows 3-8
 - debugger 3-8
 - editor 3-8
 - navigator 3-8
 - NV browser 3-8
 - protocol 3-8
 - statistics 3-8
- array 8-20
- arrow keys 3-5 *see also* keys
- authentication 6-37, 9-24, 10-6
- AUTOEXEC.BAT 4-5
- automatic build 7-3, 7-6
- automatic load 7-3, 7-6

B

- backplane transceiver 1-7, 1-8
- backplane
 - defining channels 10-18
 - development network integration 9-2
 - loading system 6-2
 - loading application node's memory 6-2
 - loading network image 6-2
- binder 11-3
- bit rate 10-20
- boot ID 7-33, 7-34
- bridge *see* routers

B

browser list 13-18, 13-20, 13-21
 see also network variables
 adding variables 13-22
 deleting variables 13-22
 modifying 13-20, 13-21, 13-22
 updating 13-23

buffer size
 protocol analyzer 4-7

BUILD.LOG file 4-12, 7-5
 commands 5-24, 5-25
 using with the editor 5-23, 5-24
 window 5-24

build all 7-3, 7-6

building custom nodes *see* nodes

build window 4-12

buttons 3-9, 3-13

C

call sequence 8-12

call stack 1-13, 8-2

channel 9-8, 10-16, 10-17

channel bit rate 9-20

C

- channel create window 10-17
 - avg packet size field 10-20
 - backplane field 10-18
 - bit rate field 10-20
 - channel name field 10-17
 - collision detection field 10-20
 - differential field 10-19
 - minimum clock rate field 10-19
 - num priorities field 10-18
 - osc accuracy field 10-20
 - powerline field 10-19
 - radio frequency field 10-19
 - single-ended field 10-19
 - special purpose field 10-19
 - transceiver type field 10-18
 - twisted pair field 10-19
- channel parameters 6-4
 - default_channel 6-4
- collision detection 10-20, 10-22
- color scheme 4-7, 4-10
- commands
 - BUILD.LOG commands
 - edit 5-25
 - next 5-25
 - previous 5-25
 - next error 5-25
 - view 5-24
 - editing commands 5-14
 - change to lowercase 5-15
 - change to uppercase 5-15
 - copy 5-15
 - copy append 5-16
 - cut 5-16
 - cut append 5-16
 - delete character left 5-16
 - delete character right 5-17
 - delete line 5-18
 - delete word left 5-18

C

commands (continued)

- delete word right 5-18
- delete to end of line 5-18
- insert control character 5-18
- insert new line 5-19
- open new line 5-19
- paste 5-19
- select block 5-14
- select block left 5-19
- select block right 5-19
- select column 5-14
- select line 5-16, 5-19
- select word 5-16, 5-20

editor commands Appendix B

movement commands 5-9

- beginning of line 5-10
- character left 5-10
- character right 5-11
- end of line 5-11
- equal indent down 5-11
- equal indent up 5-11
- line down 5-12
- line up 5-12
- match delimiter 5-12
- move cursor in window 5-10
- move text, editor window 5-5
- page down 5-13
- page up 5-13
- scroll window 5-10
- scroll window down 5-13
- scroll window up 5-13
- tab left 5-13
- tab right 5-14
- word left 5-14
- word right 5-14

prompt editor commands 5-21

- accept entry 5-21
- beginning of line 5-21
- cancel 5-21
- character left 5-21
- character right 5-21

C

- commands (continued)
 - delete character left 5-21
 - delete character right 5-21
 - delete line 5-21
 - delete to end of line 5-21
 - end of line 5-21
 - help 5-21
 - insert control character 5-21
 - paste 5-21
 - restore line 5-21
 - toggle insert mode 5-21
 - word left 5-21
 - word right 5-21
- compiler (compiling) 1-11, 1-12
 - application images 7-2, 7-3, 7-5
 - application programs 1-12
 - NEURON C programs 1-13, 7-2, 7-6
 - from the editor window 7-7
 - from the navigator window 7-7
 - warning errors 7-5
- configuration reports 11-18
 - creating 11-20
 - viewing 11-21
- configured router *see* routers
- connection information file *see* .BIF or files
- connections 11-2
 - message tag 11-5, 11-6, 11-7, 11-8, 11-10, 11-11
 - message tag connection parameter window 11-11
 - nv connection create window 11-5
 - nv connection modify window 11-5
 - nv connection parameter window 11-10
 - creating 11-5
 - deleting 11-9
 - modifying 11-7
 - parameters 11-10
 - status 11-7

C

copy text, editor window 5-5
custom network *see* network
custom nodes *see* nodes
custom transceivers *see* transceivers

D

/disk command 4-7
debugger 1-6, 1-13, 8-2
 accelerator keys 8-6
 breakpoints 8-9
 removing 8-10
 setting 8-9
 statement boundaries
 call sequence 8-12, 8-13
 changing current context 8-14, 8-15
 commands 8-6
 debugger window 8-2, 8-3
 see also debugger window
 debugging programs 8-7
 debugging code 8-8
 emulator status 8-4
 evaluating variables 8-16
 file markers 8-7
 halting running nodes 8-9
 modifying variables 8-16, 8-20
 modifying raw memory 8-23, 8-24
 modify variable dialog box 8-20
 network variables 8-26
 node status 8-4
 pointer variables 8-26, 8-27
 program error log 8-29
 raw data dialog box 8-24

D

- debugger (continued)
 - read and write protect 8-5
 - running programs 8-7
 - select data dialog box 8-17
 - single-stepping 8-11
 - source-level debugger 1-11, 8-2
 - static variables 8-28
 - timers 8-26
 - trace-stepping 8-12
 - variable display dialog boxes 8-18, 8-19
- debugger window 8-2, 8-3, 8-8
 - command button subwindow 8-2
 - source subwindow 8-2
 - status line 8-3, 8-4
- debugger application window 3-8
 - see also* application windows
- debugging nodes 2-6
- default domain 10-3
- deleting program files 5-4
- detailed build info 4-12
- developer's kit 1-12, 1-13
 - microprocessor interface developer's kit 1-14
- developer's workbench 2-2
- development cycle, LONWORKS applications 2-2

D

- development network 3-2, 9-2
 - application nodes 9-2
 - binder constraints 11-4
 - communications channels 9-2
 - connections 11-2
 - message tag 11-5, 11-6, 11-7, 11-8, 11-10, 11-11
 - message tag connection parameter window 11-11
 - nv connection create window 11-5
 - nv connection modify window 11-5
 - nv connection parameter window 11-10
 - creating 11-5
 - deleting 11-9
 - modifying 11-7
 - parameters 11-10
 - status 11-7
 - copying/modifying objects 10-29, 10-30
 - channel create window 10-17
 - channels 10-16
 - custom transceiver types 10-21
 - default domain 10-3
 - deleting objects 10-31
 - domain create window 10-12
 - domain name field 10-15
 - domains 10-12
 - emulators 9-2
 - linking nodes 11-2
 - multi-channel 10-8
 - navigator network selectors 10-9
 - network definition objects 10-10
 - network image 11-15, 11-16
 - automatic build 11-17
 - build all 11-17
 - network manager node 9-2
 - network objects 10-8
 - node modify window 10-4
 - node specifications 10-4, 10-5
 - node/variable name members 11-4
 - object definition 10-10, 10-11

D

development network (continued)

objects

copying 10-29

deleting 10-31

modifying 10-30

protocol analyzer node 9-2

SBCs 9-2

single channel 10-3

subnet 10-3, 10-14

subnet create window 10-15

transceivers, custom 10-21

see also transceivers

development station 6-26

installing target hardware 6-26, 9-18, 9-19

dialog boxes 3-6 *see also* menus

differential custom transceiver 10-21

see also transceivers

directory

application 4-4

create 4-2

home 4-2, 4-3, 4-4

project 4-2, 4-4

root 4-5

search path 4-3

structure 4-2

directory display, editor window 5-22

domain 7-11, 10-3

defining 10-12

domain create window 10-12

domain name 10-15

ID 10-13

size 10-13

domain create window 10-12

domain ID/size fields 10-13

D

DOS

- /disk command
- /exp command 4-7
- /ext command 4-7
- creating a program file from DOS 5-2
- modifying program files 5-3
- program error log 8-29
- shell 4-15

downloadable image file *see* .NXE or files

E

/exp command 4-7

/ext command 4-7

editor 1-11, 1-12

- build log 5-23, 5-24

editor accelerator key

- creating a program file 5-2
- modifying a program file 5-3, 5-4

editor application window 3-8

- see also* application windows

E

- editor window 5-3, 5-5
 - autoindent mode 5-7
 - commands
 - editing 5-14
 - movement 5-9
 - prompt editor 5-21
 - compiling NEURON C programs 7-7
 - copy 5-5
 - directory display 5-22
 - filename 5-7
 - filename extension 5-7
 - insert mode 5-7
 - macro system 5-5, 5-7
 - move 5-5
 - multiple views 5-5, 5-6
 - read-only 5-7
 - resize 5-5
 - toggle ruler line command 5-8
 - zoom 5-5, 5-8
- EEBLANK image 6-28, 7-13, 7-34, 7-36
- EEPROM 2-3, 6-10, 6-28, 6-29, 7-14, 7-32
- emulators 1-5, 1-6
 - application loading 1-6
 - debugging
 - code 8-8
 - read and write protect 8-5
 - status 8-4
 - development network 9-2, 9-3
 - installing 6-26, 7-8
 - memory read/write protection 1-6
 - reset/start/stop 1-6
 - NEURON emulators 1-13
 - source-level breakpoints 1-6
 - single-stepping 1-6
 - test commands 13-2
- ending a LONBUILDER session 4-15
- ESC key 3-5 *see also* keys
- evaluation boards 1-5, 1-10

E

expansion boards 1-3, 1-5, 1-6, 1-10

see also I/O, transceiver

exporting files 7-12, 7-16, 7-17

application image files 7-22

application image export window 7-23

components of an exported node 7-15

connection information file 7-22

building the node 7-17

creating application image 7-17

defining the node 7-17

NEURON object file 7-22

node export window 7-18

extender card 1-10

external interface file 2-5, 6-20 *see also* .XIF or files

F

F10 key 3-5

fields 3-9, 3-13

firmware

state 7-13, 7-29, 7-33

modifying firmware settings 7-38, 7-39

multiple firmware versions 7-38, 7-39, 7-40

G

generate assembly listings 4-12
generate link map 4-12
gizmo 2 multi-function I/O module 1-10

H

halt command - debugger 8-9
hardware, LONBUILDER 1-3
hardware properties 6-3, 6-6
 creating a node definition 6-6
 default_hw_props 6-3, 6-6, 6-19
 hardware properties copy window 6-7
 firmware version field 6-9
 HW property name field 6-7
 input clock rate field 6-8
 NEURON CHIP field 6-8
 NEURON CHIP firmware field 6-8
hardware properties copy window 6-7
hardware test results window 13-6, 13-7
 error fields 13-7
 lost messages field 13-7
 missed messages field 13-8
 rcv transaction full field 13-8
 ROM s/w version field 13-7
 transmission timeouts field 13-8
help 3-19, 3-20
HEXCNVRT.EXE file 7-34, D-2
home directory 4-2, 4-3, 4-4
host C definition file 6-20
 see also .HC file

IDE *see* integrated development environment

image 6-4, 6-20, 11-15 *see also* application,
node, development network

images directory 7-39

importing files 7-12

 application image create window 6-21

 application image files 7-25

 app image name 7-21, 7-25

 app image origin 7-21, 7-25

 external interface files 7-21

include file 4-14

 deleting 5-4

 modifying 5-3

installing custom nodes *see* nodes

integrated development environment 1-11, 1-12

 editor 1-12

 navigator 1-12

 object database 1-12

 project manager 1-12

 tools 1-12

interface adapter 1-3, 4

I/O 1-5

 hardware 1-6

 off-chip memory 6-10

I/O interface 1-5

 prototypes 1-10

I/O port start address 4-7, 4-10

K

keys *see also* menu
 accelerator 3-5, Appendix A
 arrow keys 3-5
 ESC key 3-5, 3-6

L

lb command 4-6
LBCHECK D-3
LBDBCHK D-3
LBDBFIX D-4
LBTEST D-4
lbe command 5-2
learning router *see* routers
LIM 4.0 4-8
link maps 4-12
linking nodes 11-2

L

LONBUILDER

- application interface board 1-10
- application programming tools 1-11
 - see also* application programming tools
- configuring project parameters 4-11
 - see also* project parameters
- development station 1-3, 7-28
- developer's workbench 1-2, 1-3
- editor 7-40
- ending a session 4-15
- expansion boards 1-3
- extender card 1-10
- IDE 7-38, 7-39, 7-40
- interface adapter 1-3, 1-4
- I/O evaluation boards 1-5
- microprocessor interface developer's kit 1-14
- multi-function I/O kit 1-10
- network management tools 1-11, 7-28
 - see also* network manager
- network manager 1-14 *see also* network manager
- NEURON emulator 1-6 *see also* emulator
- processor boards 1-3
- protocol analyzer 1-14 *see also* protocol analyzer
- RF transceiver 1-9 *see also* transceiver
- router 1-7 *see also* router
- single board computer (SBC) 1-6
 - see also* single board computer
- starting a session 4-6
- start-up screen 3-2 *see also* start-up screen
- software tools 1-11 *see also* software tools
- system directory 4-13
- TP-RS485 transceiver 1-9 *see also* transceiver

L

LONMANAGER API 7-14

LONPROJ 4-5

AUTOEXEC.BAT 4-5

environment variables 4-5

starting a LONBUILDER session

LONTALK

packets 4-9

protocol 1-2, 1-14

interoperability 6-32

SNVTs 6-32

serial adaptor 1-14

LONTALK API 7-14

LONWORKS

application development cycle 2-2

see also application development cycle

application prototyping 1-10

development cycle 2-2

network 1-14

TP/XF-78 1-8

TP/XF-1250 1-8

M

macro system, editor window 5-5

main menu bar 3-3

memory map 1-6, Appendix C

NEURON 3150 CHIP memory map sample C-2, C-3

NEURON 3120 CHIP memory map sample C-5, C-6

M

- memory properties 6-10
 - 3120 chip memory allocation 6-11
 - 3150 chip memory allocation 6-10, 6-12, 6-13
 - configuring off-chip memory 6-10, 6-12
 - memory properties window 6-12
- memory properties window 6-12
 - EEPROM end field 6-14
 - EEPROM size field 6-13
 - EEPROM start field 6-14
 - EEPROM write time field 6-13
 - I/O end field 6-15
 - I/O size field 6-15
 - I/O start field 6-15
 - RAM end field 6-14
 - RAM size field 6-14
 - RAM start field 6-14
 - ROM size field 6-13
 - ROM start field 6-13
 - ROM end field 6-13
- menu (s) *see also* windows
 - accelerator keys *see* accelerator keys
 - bar 3-3, A-2
 - commands Appendix A
 - dialog boxes 3-6, 3-7, 3-9
 - invoking commands 3-5
 - pull-down 3-4, 3-5, 3-6, 3-8
 - opening 3-4
 - pointer 3-4
 - submenus 3-6
 - windows 3-8, 3-9
 - buttons 3-9, 3-13
 - fields 3-9, 3-13
 - object lists 3-9, 3-10
 - scroll bars 3-9, 3-13
- message tags 11-5, 11-6, 11-7, 11-8, 11-10, 11-11

M

- message tag connection parameter window 11-11
 - auto field 11-12
 - domain 11-14
 - manual field 11-12
 - priority field 11-14
 - receive timer field 11-13
 - repeat timer field 11-13
 - retries field 11-13
 - service type field 11-14
 - transaction timer field 11-12
 - transport layer messaging parameters 11-11

- microprocessor interface developer's kit 1-14

- microprocessor interface program 1-14, 2-6

- MIP *see* microprocessor interface program

- modifying raw memory 8-23

- modify variable dialog box 8-20

- mouse 3-4

- movement commands 5-9

- multi-function I/O kit 1-10

- multiple views, editor window 5-5, 5-6

N

navigator 1-12, 3-17, 3-18

navigator application window 3-8
see also application windows

navigator window
 compiling NEURON C programs 7-7
 creating a program file 5-2, 5-3
 modifying a program file 5-3, 5-4

navigator home window 3-2, 3-3

network

 addressing 1-15 *see also* network image
 channels 9-8
 connections 1-15, 11-2 *see also* development networks
 design 2-8
 development 9-2
 see also development network
 image 2-2, 2-3, 11-15 *see also* development networks
 loading over 6-2
 multi-channel topologies 9-8
 topology errors 9-9
 objects 10-10, 10-29
 performance 1-15
 production 9-2
 routers 9-8 *see also* routers
 target hardware 9-10, 9-11
 subnet 7-11, 9-11
 traffic 1-15
 variables 2-5 *see also* network variables

network configuration, building 1-12

network image 1-15, 2-3, 7-2

network management tools 1-11, 1-14
 network manager 1-14
 protocol analyzer 1-14

N

- network manager 1-11, 1-14, 1-15, 7-28, 7-29
 - commands 13-15
 - loading application image 2-7
 - loading nodes and routers 7-8
 - network installation 9-2
 - NEURON ID 7-36
 - installing nodes 2-8
 - test commands 13-2 *see also* test commands
 - testing custom nodes 2-7
 - wink messages 13-12

- networks *see* twisted pair, radio frequency

- network statistics window 12-3
 - error statistics field 12-5
 - network field 12-5
 - performance statistics field 12-4
 - session field 12-4
 - start time field 12-3
 - transport field 12-4
 - update time field 12-3

- network variable browser *see* network variables

- network variable connection create window 11-5

- network variable connection modify window 11-8

- network variable connection parameters window 11-10

N

- network variables 2-5, 6-32
 - browser list 13-18, 13-20, 13-21
 - adding variables 13-22
 - deleting variables 13-22
 - modifying 13-20, 13-21, 13-22
 - updating 13-23
 - browsing 13-16
 - connections 11-4, 11-5, 11-6, 11-7, 11-9
 - creating 13-17
 - deleting 13-17
 - debugger 8-25
 - display variable dialog box 13-19
 - editing 6-35
 - evaluating 8-16
 - select data dialog box 8-17
 - importing definitions 6-34
 - importing SNVT information 6-39, 7-14
 - inputs 11-2
 - message tag connection parameter window 11-11
 - modifying 8-16, 8-20, 8-21, 8-22
 - name members 11-3
 - nv browser window 13-19
 - nv connection create window 11-5
 - nv connection modify window 11-8
 - nv connection parameters window 11-10
 - node query window 6-32
 - outputs 11-2
 - query command 7-14
 - querying information 6-33
 - self-documentation 6-32
 - self-identification 6-32
 - target network 7-29
 - viewing parameters 6-36
- NEURON 3120 CHIP 1-6, 1-13
 - building custom-based nodes 7-13, 7-26
 - guidelines 7-27
 - configuring project parameters 4-12
 - memory allocation 6-10
 - memory map C-5, C-6
 - programming 3120 chip memory 7-35
 - system image 6-2

N

- NEURON 3150 CHIP 1-6, 1-13
 - configuring project parameters 4-12
 - building custom-based nodes 7-13, 7-26
 - guidelines 7-27
 - exporting .NRI files 7-13
 - loading 7-37
 - memory allocation 6-10
 - memory map C-3, C-4
 - recovering with EEBLANK image 6-28
 - system image 6-2
- NEURON C file
 - deleting 5-4
 - modifying 5-3
- NEURON C compiler *see* compiler
- NEURON C debugger 1-6, 8-2
 - see also* debugger
- NEURON C debugger window
 - see* debugger window
- NEURON C developer's kit 1-12, 1-13
- NEURON C programs
 - compiling 7-6
- NEURON C source file 4-12, 6-20 *see also* .NC file
- NEURON CHIP 1-2
 - EEPROM 2-3 *see also* EEPROM
 - firmware 2-2 *see also* firmware state
 - input clock 7-36
- NEURON EEPROM image file *see* .NEI or files
- NEURON ID 7-36
- NEURON object file *see* .NO or files
- NEURON ROM image file *see* .NRI or files

N

node (s)

- addressing 1-15
- binder constraints 11-4
- browsing memory 13-13, 13-14
- browsing network variables 13-16
 - browse 13-17
 - create 13-17
 - delete 13-17
- configuration reports 11-18
 - creating 11-20
 - viewing 11-21
- connections 11-2 *see also* development networks
- custom 2-8, 7-26, 7-27
 - firmware state 7-29, 7-33
 - installing 7-36
 - loading 7-37
 - network types 7-29
 - programming 7-31, 7-32
 - programming 3120 chip memory 7-35
- define external interface 2-5
- define image 6-20
- domain/illegal domain 7-11
- exporting 7-12, 7-15 *see also* exporting files
- firmware 7-38 *see also* firmware
- function assignment 2-5
- hardware properties 6-12
- identification 2-5
- image 6-4, 6-20
- importing 7-12 *see also* importing files
- installing 2-8, 6-26
- linking 11-2
- loading 7-8
 - automatically 7-9
 - manually 7-9
 - through routers 7-10
- name members 11-3
- network management 9-5
 - changing node definition 9-5
 - commands 13-15
 - installing on target hardware 9-6

N

- node (continued)
 - node export window 7-18
 - node modify window 10-4
 - node query window 6-33, 6-34
 - objects, 6-30
 - deleting 6-31
 - modifying 6-30
 - off-line 1-15, 13-10
 - on-line 1-15, 13-10
 - protocol analyzer 9-5
 - changing node definition 9-5
 - reset 13-11
 - specifications 6-2, 6-4, 6-23, 9-12, 9-22, 10-4, 10-5
 - target hardware 6-26
 - target hardware memory window 13-14
- node configuration reports 11-18
 - creating 11-20
 - viewing 11-21
- node definitions 3-2
- node export window 7-18
 - applicationless option 7-19
 - base file name field 7-18
 - configured option 7-20
 - file format options 7-19
 - file types field 7-19
 - firmware state options 7-19
 - node export dialog box 7-18
 - unconfigured option 7-20
- node memory
 - application image 2-2
 - network image 2-2
 - system image 2-2

N

- node modify window 10-4
 - app image name 10-5
 - authentication fields 10-6
 - hw name field 10-7
 - location field 10-7
 - max free buffer wait field 10-6
 - node name field 6-25
 - subnet name fields 10-5
 - subnet/node number fields 10-5
- node query window 6-33, 6-34, 6-35
 - array size field 6-38
 - authentication field 6-37
 - avg message rate 6-38
 - change only if offline field 6-36
 - config class field 6-37
 - config/no config field 6-37
 - direction field 6-37
 - max message rate 6-38
 - network variable name field 6-38
 - polled field 6-36
 - priority field 6-37
 - sd string for netvar field 6-38
 - service type field 6-37
 - synchronous field 6-36
- node specification create window 6-24
 - app image name field 6-25
 - node name field 6-25
 - target HW field 6-25
- num priorities
 - defining channels 10-18
- NV browser application window 3-8
 - see also* application windows
- NVRAM 7-14, 7-19

- object
 - copying 10-29,
 - definitions 9-12, 10-10
 - deleting 10-31
 - modifying 10-31
- object code files 6-20
 - see also* .BIF file, .NO file
- object database 1-12, 3-2, 4-4, 6-4, 6-5
 - application node objects 6-30
 - building 7-2
 - deleting 6-31
 - modifying 6-30
 - node specification 6-23
 - target hardware objects 6-16
- object lists 3-9, 3-10, 13-2 *see also* windows
- off-chip memory
 - EEPROM 6-10, 6-12, 6-13
 - I/O 6-10, 6-12
 - RAM 6-10, 6-12
 - ROM 6-10, 6-12
- options menu 4-7
- oscillator accuracy 10-20
- overlay memory 4-7, 4-8
 - /disk command 4-7
 - /exp command 4-7
 - /ext command 4-7

P

- packet buffers 9-17
- packet size 10-20
- powerline transceiver 10-19, 10-21
 - see also* transceivers
- power supply, external 106
- processor boards 1-3, 1-5, 1-10
 - LONBUILDER NEURON emulators 1-5
 - see also* emulators
 - LONBUILDER routers 1-5, 1-7
 - see also* routers
 - LONBUILDER single board computers (SBC) 1-5
 - see also* single board computers
- program error log 8-29
- program files, creating 5-2
 - DOS 5-2
 - editor accelerator key 5-2
 - lbe 5-2
 - navigator window 5-3
 - powerline 10-19
- program files, deleting 5-4
- program files, modifying 5-3
- programming custom nodes *see* nodes
- project configuration window 4-13
- project directory 4-3, 4-4
- project manager 1-11, 1-12, 7-2
 - application directories 4-4
 - errors 7-5
 - warnings 7-5
 - window 7-4
- project manager window 7-4, 7-7
 - invoke editor command 7-5

P

- project parameters 4-11
 - application directories field 4-14
 - automatic build 4-13
 - build all 4-13
 - build options 4-12
 - build.log file 4-12
 - compile 4-11
 - detailed build info 4-12
 - generate assembly listings 4-12
 - generate link map 4-12
 - include directories field 4-13
 - invoke editor on build stop 4-13
 - load after build 4-13
 - load 4-11
 - link 4-11
 - modify defaults 4-14
 - output link summary 4-12
 - stop build 4-13
 - don't stop 4-13
 - errors 4-13
 - stop after compile 4-13
 - warnings 4-13
 - stop after load 4-13
- prom programmer 7-12, 7-13
- prompt editor 5-21

P

- protocol analyzer 1-14, 1-15
 - collecting message packets 12-6
 - disk full condition 12-7
 - log filter dialog box 12-8
 - message interpretation 12-17
 - more window for requests 12-18
 - more window for responses 12-19
 - network traffic statistics 12-2
 - packet logs 12-6, 12-7, 12-8
 - adding message tags 12-9
 - adding network variables 12-9
 - adding nodes 12-9
 - creating 12-6
 - destination nodes 12-9
 - filter/filtering 12-8, 12-10, 12-11, 12-12, 12-14
 - packet types 12-9
 - protocol analyzer window 12-14, 12-15, 12-16
 - source nodes 12-9
 - turning on 12-7
 - variables/tags 12-10
 - viewing 12-12
 - protocol analyzer window 12-14, 12-15, 12-16
 - statistics window 12-3
 - viewing message packets 12-6
- protocol analyzer buffer size 4-7, 4-9
 - storing LONTALK packets 4-9
- protocol application window 3-8
 - see also* application windows
- prototyping 1-10
- pull-down menus 3-3 *see also* menus

R

radio frequency networks 1-5, 1-7, 9-8

RAM, off-chip memory 6-10

raw data dialog box 8-24, 8-25

raw memory

evaluating 8-23

modifying 8-23

read and write protect map 8-5

repeater *see* routers

resize, editor window 5-5

RF networks *see* radio frequency networks

ROM, off-chip memory 6-10

root directory 4-5

router hardware create window 9-14

channel name field 9-16

clock rate field 9-17

location field 9-15

memory field 9-16

NEURON id field 9-17

node specs field 9-15

packet buffers field 9-17

priority field 9-16

router hw name field 9-14

router hw type field 9-15

size field 9-17

R

- routers 1-5, 1-7, 9-4
 - authentication 9-24
 - channel definition 9-13
 - defining node specifications 9-22, 9-23
 - domain/illegal domain 7-11, 10-11
 - installation 9-19, 9-20, 9-21
 - loading 7-8
 - automatically 7-9
 - manually 7-10
 - location 9-25
 - multi-channel topologies 9-8
 - network management authentication 9-25
 - node specifications 9-13
 - object definitions 9-12, 10-11
 - off-line 13-11
 - on-line 13-11
 - project manager 9-21
 - reset 13-12
 - router hardware create window 9-14, 9-22
 - routing messages 9-8
 - side A 9-13
 - side B 9-13
 - subnet 7-11, 9-11, 9-23, 9-24
 - target hardware 9-10, 9-11, 9-12, 9-13
 - installing in a development station 9-18, 9-19
 - installing on a remote network 9-19
 - name 9-24
 - testing 13-9
 - types
 - bridge 1-7, 9-8
 - configured 1-7, 9-8
 - learning 1-7, 9-8
 - repeater 1-7, 9-8, 9-15, 9-19

S

SBC

see single board computers

select data dialog box 8-17, 8-23

self-documentation 6-32

self-identification 6-32

serial LONTALK adapter 1-14

scroll bars 3-9, 3-13, 3-14

single board computers (SBC) 1-5, 1-6, 1-7

based network 7-28

development networks 9-3

installing 6-26

installing on a remote network 6-27

test commands 13-2

testing application code 2-7

single-ended custom transceiver 10-21

see also transceivers

single-stepping 1-13, 8-11

SNVT *see* standard network variable types

software tools 1-11

source level debugger 1-11

special purpose custom transceiver 10-21

see also transceivers

standard network variable types 6-32, 6-39

see also network variables

installing 7-8

query command 7-14

starting a LONBUILDER session 4-6

lb command 4-6

start-up screen 3-2

status line 3-3

S

- statistics application window 3-8
 - see also* application windows
- static variables 8-28
- status line 3-3
- submenus 3-6 *see also* menus
- subnet 7-11, 9-11, 10-3, 10-14
 - subnet create window 10-15
- subnet create window 10-15
 - domain name field 10-15
 - subnet number field 10-15
- system command 4-7
- system image 2-2, 2-3
- system parameters 4-7
 - color scheme 4-7, 4-10
 - I/O port start address 4-7, 4-10
 - options menu 4-8
 - overlay memory 4-7, 4-8
 - protocol analyzer buffer size 4-7, 4-9
 - system command 4-8
 - system parameters window 4-8
- system parameters window 4-8

T

- target hardware 6-4, 6-16
 - creating objects 6-16
 - create command 6-16
 - copy command 6-16
 - EEPROM memory 6-29
 - installing 6-23
 - custom node 6-27
 - emulator 6-23
 - network management nodes 9-1, 9-6, 9-7
 - remote network 6-27
 - SBC 6-23
 - objects 6-16
 - emulator_1 6-16
 - emulator_2 6-16
 - reset 13-11
 - target hardware window 13-2
 - target hw copy window 6-17
 - testing
 - node hardware status 13-3, 13-14
 - target hardware location 13-3
 - target hardware status 13-3
 - target hardware window 13-2
- target hw copy window 6-17
 - app hw name field 6-17
 - channel field 6-18
 - collision detect field 6-19
 - HW prop name field 6-19
 - HW type field 6-18
 - location field 6-18
 - NEURON ID field 6-19
 - node specs field 6-19
 - priority field 6-19
 - object list 13-2
- target hardware memory window 13-14

T

- test commands 13-2
 - application image status 13-4, 13-5
 - emulator status 13-4
 - hardware status 13-3, 13-4, 13-5
 - location 13-3
 - node hardware status 13-3
 - SBC status 13-4, 13-5
 - status window 13-2
- testing nodes 2-6, 2-7, 13-6 *see also* debugging nodes
 - application nodes 13-6
 - hardware test results window 13-6, 13-7
 - test commands 13-2
 - application image status 13-4, 13-5
 - emulator status 13-4
 - hardware status 13-3, 13-4, 13-5
 - location 13-3
 - node hardware status 13-3
 - SBC status 13-4, 13-5
 - status window 13-2
 - testing routers 13-9
- test results window 13-6, 13-7
 - see also* hardware test results window
- time factors window 10-26
- toggle ruler line command, editor window 5-8
- topology errors 9-9, 9-10
- trace-stepping 8-12

T

- transceiver 1-2, 1-5, 9-20
 - see also* individual transceiver types
 - custom transceivers 1-5, 10-21
 - alternate rate 10-25
 - bit rate 10-22
 - bit sync threshold 10-23
 - collision detection 10-22
 - differential 10-21
 - filter 10-23
 - general purpose data 10-25
 - hysteresis 10-23
 - indeterminate time 10-27
 - interface rate 10-24
 - minimum interpacket 10-27
 - preamble length 10-26
 - raw data 10-28
 - receive start/end delay 10-27
 - single-ended 10-21
 - special purpose 10-21
 - wakeup pin dir 10-26
 - xcvr controls preamble 10-24
 - defining channels 10-18
 - differential 10-19
 - filter 10-23
 - hysteresis 10-23
 - powerline 10-19
 - radio frequency 10-19
 - single-ended 10-19
 - special purpose 10-19
 - alternate rate 10-25
 - general purpose data 10-25
 - interface rate 10-24
 - wakeup pin dir 10-26
 - xcvr controls preamble 10-24
 - twisted pair 10-19
 - type 10-18
- expansion boards 1-7

T

LONBUILDER RF transceiver 1-9
LONBUILDER TP-RS485 1-9
time factors window 10-26

TSR 4-15

twisted-pair 1-5, 1-7, 1-8, 10-21
see also transceivers
LONWORKS TP-RS485 1-9
performance 1-8, 1-9

U

utilities Appendix D
HEXCNVRT D-2
LBCHECK D-3
LBDBCHK D-3
LBDBFIX D-4
LBTEST D-4

V

variable display dialog box 8-18
variables *see* network variables

W

windows 3-9 *see also* menus
 buttons 3-9
 fields 3-9
 object lists 3-9, 3-10, 3-23
 scroll bars 3-9

wink command 13-12

Z

zoom, editor window 5-5, 5-8